VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Internetové rádio

Internet radio

2010

Daniel Kapča

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava*.

V Ostravě 7. května 2010 .....

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. května 2010 .....

Rád bych na tomto místě poděkoval všem, kteří mi s prací pomohli, protože bez nich by tato práce nevznikla. Obzvláště chci poděkovat svému kamarádovi Radimovi, který mě na téma této práce přivedl. Zvláštní poděkování patří testerům, kteří si dali tu práci a nervy s testováním a hlášením nedostatků.

Abstrakt

Tato práce pojednává o tvorbě internetového rádia s využitím dostupných technologií a protokolů. Ukáže hlavní problematiku tvorby serverové a klientské části a jejich řešení. Dále popisuje využití nejrůznějších technologií pro zaznamenání, kompresi a distribuci audio dat.

Klíčová slova: diplomová práce, rádio,klient, server, VST, RTSP, SHOUTCAST, metadata, protocol, broadcast

Abstract

This piece of work is about making of internet radio with use available technologies and protocols. Work show main problems in making server and client parts with solution. Also work describe use of various technology for audio data recording, encoding and distribution.

Keywords: master thesis, radio, client, server, VST, RTSP, SHOUTCAST, metadata, protocol, broadcast

Seznam použitých zkratek a symbolů

NGS	–	Next Generation Streaming
RTSP	–	RealTime Streaming Protocol
VST	–	Virtual Studio Technology
VAC	–	Virtual Audio Cable
FLAC	–	Free Lossless Audio Codec
IETF	–	Internet Engineering Task Force
RFC	–	Request for Comments
DAW	–	Digital Audio Watermarking
LAME	–	Knihovna pro převod MP3
PCM	–	Pulse Code Modulation
RAW	–	Nezpracovaná data
MIME	–	Multipurpose Internet Mail Extension

Obsah

1 Úvod

1.1 Pár slov na začátek

Rádio je nedílnou součástí lidí již nějaké to desetiletí a neustále si drží svojí přízeň i v dnešní době. Jistě každý alespoň jednou zatoužil mít svojě rádio a ukázat světu svou hudební vizi. Naneštěstí rádiová technika je neustále velmi nákladná a získat vysílací právo na frekvenci je velký problém. S příchodem, ale hlavně masivním rozšířením internetu se tyto dva hlavní problémy odstranily. Naneštěstí se objevily nové, ale ty nejsou až tak markantní jako u klasického analogového typu. Mezi nové problémy patří například konektivita (rychlost, velikost objemu přenesených dat) nebo hardwarové požadavky. Dalším problémem je bohužel i rapidní nedostatek kvalitního softwaru pro provoz rádio serveru (i když se samozřejmě najdou). Problémy radiového software můžeme v zásadě rozdělit na několik kategorií:

- Příliš komplikovaný
- Příliš jednoduchý
- Příliš cílený
- Uživatelsky nepřívětivý
- Softwarově vázaný

Příliš komplikovaný - Takový software většinou poskytuje širokou paletu možností, ale za cenu nepřehlednosti. Běžný uživatel bez odborného zaškolení se v něm prakticky ihned ztratí a velmi dlouho se v něm musí orientovat. Velkou výhodou těchto softwaru je intergrace reklam do vysílaní a jsou tedy určeny spíše pro komerční stanice.

Příliš jednoduchý - Tento software je určen převážně pro běžné uživatele, kteří si chtějí zkusit vysílat jen tak pro zábavu. Poskytuje jen základní rozhraní a moc toho neumí.

Příliš cílený - Jak vyplynulo z předchozích dvou kategorií, tak většina dnešního softwaru je příliš cílená na specifické skupiny uživatelů a nejsou dostatečně univerzální.

Uživatelsky nepřívětivý - Velkým nedostatkem dněšních aplikací ať už radiových nebo ostatních je uživatelská nepřívětivost. Existuje rčení, že „Vzhled prodává“ a u softwaru to platí také, ať už se jedna čistě jen o samotný vzhled aplikace nebo třeba jen o uspořádání ovladacích prvků.

Softwarově vázaný - Hlavní nedostatek stávajících rádiových aplikací. Nutí zakoupit dodatečný software nebo mají integrované funkce, které většinou nedostačují. Výhodou použití „vnitřních“ řešení je plná podpora různých funkcí, které nelze snadným způsobem zajistit při použití softwarů třetích stran (například názvy skladeb).
Internetové vysílání se dělí na dva typy.

1.2 Podcast

Prvním typem je tzv. „Podcast“. Podcastem se nazývá druh neživého vysílání, které někdo dopředu předtočí, uloží ho jako audio soubor někde na server a případný posluchač

si ho může poslechnout. Výhodou tohoto druhu vysílaní je možnost ovládání přehrávání. Nahrávka může být většinou (když je podporovana serverem a protokolem) pozastavena, přetočena, posunuta dopředu nebo dozadu apod. Další výhodou je možnost opakovaného poslechu, který je u běžného vysílání nemožný. Podcastovány mohou byt jak audio, tak i video nahrávky. V podstatě je to obyčejný mediální soubor jako např. mp3.

1.3 Livecast

Druhým typem je tzv. „Livecast“. Livecastem se nazývá druh živého (jak lze dle názvu usoudit), které je jako u běžného rádio živě vysíláno. Nějaké speciální výhody jako Podcast ovšem nemá. Má ovšem jednu velkou nevýhodu a tou je kvalita přenosu. Při velké kvalitě vysílaní vzniká velký datový tok a uživateli nemusí stačit jeho internetové připojení (at' už rychlostí nebo omezením objemu pro přenos dat). Tento problém se dá eliminovat použitím různých kodeků pro kompresi nebo snížením kvality vysílaní (ovšem za cenu horšího poslechu). Stejně jako u Podastu, tak i u Livecastu se mohou vysílat jak audio, tak i video nahrávky. Pro jejich zaznamenání a uložení je potřeba specialního software.

1.4 DJ's

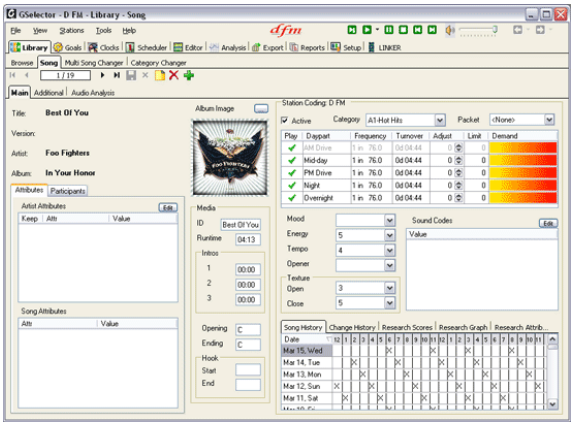
Nedílnou součástí radií jsou DJ's, kteří vesměs udávají hudební směr rádia. Z nějaké důvodu rádiové aplikace na tyto lidi zapomínají a něvenují jim dostatečnou pozornost. Pokud pomineme možnost, že DJ hraje z playlistu (seznam skladeb), který podporuje 99% aplikací, tak autoři zapomínají na živé hraní, jak z PC, tak z externích zařízení. Tohle vidím jako největší nedostatek stávajících aplikací.

2 Aktuální stav na trhu

Na trhu se vyskytují 2 hlavní druhy softwaru pro rádiové vysílaní. Prvním druhem jsou tzv. Schedulery - Plánovače, pomocí kterých se plánuje vysílání. Mezi ně patří například GSelector[3]. Druhým druhem jsou Broadcastery - Vysílače. Tyto naopak vysílají audio data na internet mezi posluchače. Mezi ně patří například Shoutcast DSP Winamp plugin[4] nebo Icecast server[5].

2.1 GSelector

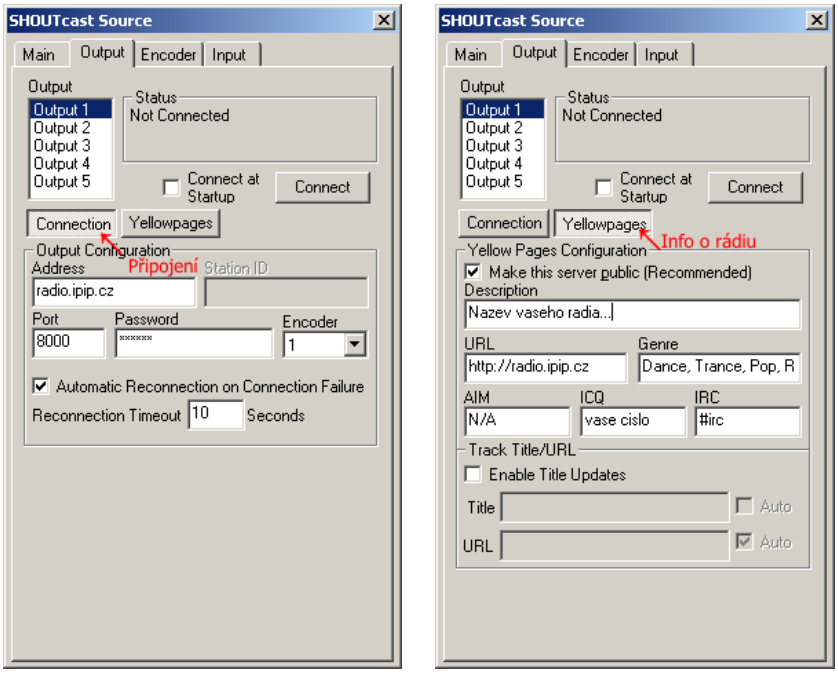
GSelector patří je plánovací systém, ve kterém se dají vytvářet vysílací playlisty a celkově spravovat vysílání. Vyniká spoustou funkcí, které dokáží zlepšit kvalitu vysílání. Mezi ně patří různé statistiky, vybírání vhodných skladeb, zařazování neplánovaných skladeb do volných pozic nebo kontrola přehrávání, zda-li například nehraje stejná skladba na více stanicích.



Obrázek 1: Obrazovka programu GSelector

2.2 Shoutcast DSP Winamp plugin

Mnohem jednodušší a hlavně pro amatéry je určen tento plugin. Winamp je jeden z nejznámějších audio přehravačů. Svou oblibu si získal hlavně díky jednoduchosti a rozšířitelnosti. A právě zde přichází na řadu Shoutcast DSP Winamp plugin. Tento plugin využívá také rádiový server Radioo[6]. Na server Radioo beží tzv. Icecast server (viz. sekce 2.3) a klient pomocí přehravače Winamp přehrává požadovanou hudbu. Tuto hudbu plugin zachytí, překóduje na zvolený formát a odešle na Radioo Icecast server (tento server není napevno zvolený).



(a) Nastavení připojení (b) Nastavení rádia

Obrázek 2: Ukázka nastavení Shoutcast DSP pluginu

2.3 Icecast

Již zde padl pojem Icecast. Icecast [5] je bezplatná technologie streamovacího serveru, který je multiplatformní a využívá ho docela velké množství programů a webů. Mezi významnější aplikaci, která ho využívá patři Traktor Dj Studio, který má přímo integrovanou podporu pro přímé vysílání. Server je konzolová aplikace a podporuje formáty mp3 nebo ogg včetně několika druhů playlistů (např. m3u).

3 Technologie

Dříve, než se pustím do popisu samotné tvorby rádia vás musím seznámit s technologiemi, které v něm budou použity (anebo jsou alespoň plánovány). Některé technologie podléhají **Standardům**. Standardy v internetových technologiích ma na starosti organizace IETF [1]. Samotné standardy jsou uvedeny jako RFC [2] dokumenty ve tvaru rfcNNNN, kde N je číslo (např. rfc2616 je standard pro HTTP/1.1).

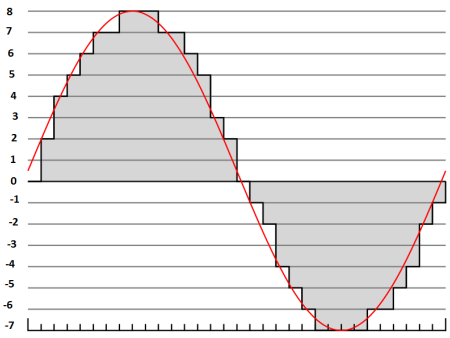
3.1 Základ zpracování audia na počítači

Abychom mohli zaznamenávat a zpracovávat digitální zvukový záznam, musíme znát základní formát a strukturu, v jakém je digitální zvuk zpracováván. Základní zvukovou jednotkou je Vzorek (Sample), který se skládá z kanálů (např. pro Stereo ze 2 kanálů), které mají bitovou hloubku (nejběžněji 16 bitů).

Obrázek 3: Ukázka formátu vzorku (2 vzorky: Stereo, 16 bit)

Velikost vzorku se určuje podle velikosti a počtu kanálů. Při standardním nastavení kvality (bitové hloubky) 16 bitů (2 byty) na kanál a 2 kanálech (stereo) má jeden vzorek 4 byty (2 byty levý kanál, 2 byty pravý kanál).

Vzorek představuje aktuální část průběhu sinusoidy. Z matematiky nebo fyziky víme, že sinusoida má interval $\langle -1; 1 \rangle$. Bitová hloubka nám určuje přesnost průběhu sinusoidy.



Obrázek 4: 4 bitová hloubka sinusoidy

Při běžné bitové hloubce 16 bitů se jako datový typ používá short. Short má rozsah $\langle -32767; 32768 \rangle$, to znamená, že rozsah sinusoidy $\langle -1; 1 \rangle$ můžeme rozdělit na 65536 dílů. Bežně se používají datové hloubky 4, 8, 16 a 24. Všechny tyto datové hloubky mají jednu nepříjemnou vlastnost => jsou to celočíceselné datové typy. Jejich nepříjemnou vlastností

je, že nemají dostatečnou přesnost a sinusoida není zcela plynulá (jak lze vidět např. na obrázku 4). S příchodem Windows Vista a nové generace zvukových karet se začal používat 32 bitová float point (s plovoucí desetinnou čárkou) hloubka. V této hloubce se používá rozsah $\langle -1; 1 \rangle$ pomocí kterého získáme nejpřesnější a nejjemnější rozlišení zvuku. Naneštěstí ne všechny aplikace tuto hloubku neumí využívat a provádí převod do celočíselných datových typů (obvykle 16 nebo 24).

Posledním velmi důležitým pojmem je Vzorkovací frekvence. Ta určuje počet vzorků za sekundu. Standardně se používá vzorkovací frekvence 44100 nebo 48000. Nyní ovšem nastává jedna nepříjemná vlastnost. Pokud chceme mít např. 10 vteřin stereo zvuku o vzorkovací frekvenci 48000 a 16 bitové (2 byty) hloubce na kanál, tak získáme:

$$48000 * 2 * 2 * 10 = 1920000 = 1,92MB$$

Pro pouhých 10 vteřin audio záznámu získáme téměř 2 MB dat, což je obrovská zátěž jak pro přenost po sítí, tak i pro uložení na záznamová média. Pro snížení této náročnosti byly vyvinuty různé kompresní algoritmy (kodeky), které tyto velikosti snižují. Máme 2 druhy kodeků: ztrátové a bezztrátové.

Ztrátové kodeky (např. MP3) jsou kodeky, které zvuková data komprimují odstraněním pro bežné lidské ucho neslyšitelných frekvencí. Tím pozmění původní originální zvuk a nelze ho již nikdy přesně dekódovat zpět. Nevýhodou těchto kodeků je ta, že mají různé kvality a mohou způsobovat různé zvukové efekty nebo snížit zvukovou kvalitu. Na druhou stranu dokáží poměrně významně snížit velikost nahrávky.

Bezztrátové kodeky (např. FLAC) jsou kodeky, které zvuková data komprimují pomocí algoritmů, které neničí původní nahrávku a je možné tedy dekódovat zpět do původní podoby. Jejich nevýhodou je, že nedokáží zkomprimovat nahrávku na tak malé velikosti jako ztrátové kodeky. Jejich výhodou je však zachování původní kvality nahrávky.

Více o různých kodecích je popsáno v kapitole 3.3.

3.2 Zvukové technologie

Internetové rádio je postavené na zvukových technologiích a je to jeden z klíčových prvků celého projektu. V této části popíšu aktuálně nejpoužívanější technologie, které jsou dostupné na trhu.

3.2.1 BASS

BASS [12] je zvuková knihovna pro Windows a MacOSX. Poskytuje API pro přehrávání a zaznamenávání zvuku s využitím DirectX (minimálně vyžaduje DirectX 3), MMX instrukcí, hardwarové akcelerace a případně ASIO ovladačů. Pro správné fungování je nutná přítomnost zvukové karty (alespoň virtuální). Tato karta (ovladač) nemusí nutně podporovat DirectX rozhraní, ale omezí se tím výkon zpracování. Mezi zajímavé vlastnosti BASS patří:

- Podpora WAV/AIFF/MP3/MP2/MP1/OGG
- Podpora Internetové streamování
- Podpora vícekanálového zvuku
- Add-on systém
- Podpora zaznamenávání
- Podpora 3D zvuku včetně EAX
- Podpora 32 bitového zpracování

Samotné rozhraní BASS je napsáno v C++ (nyní ve verzi 2.4.5) a existuje přímý wrapper pro C#, Javu a Flat Assembler. Pro celou knihovnu existuje vynikající dokumentace, která ve verzi pro C# může být integrováná přímo do Visual Studia. K této dokumentaci je řada ukázkových příkladů a velkou uživatelskou základnu.

BASS projekt obsahuje poměrně velké množství Add-Onů. Mezi ně patří například:

- **BASSWMA** - Podpora pro formát Windows Media Audio
- **BASSenc** - Podpora command line převaděčů jako LAMEENC nebo OGGENC
- **BASSmix** - Možnost mixování kánálů do jednoho nebo provádění převzorkování
- **BASS_VST** - Podpora VST pluginů
- **BASSASIO** - Podpora ASIO rozhraní
- Podpora pro různé audio formáty

BASS nabízí 4 licenční politiky.

- **Zdarma** - Pro nekomerční aplikace
- **100 €** - Pro libovolný počet aplikací, kdy každá z nich musí mít cenu do 40 Euro
- **950 €** - Pro jednu čistě komerční aplikaci
- **2450 €** - Pro libovolný počet čistě komerčních aplikací

3.2.2 OpenAL

OpenAL (Open Audio Library) je multiplatformní 3D zvuková knihovna od společnosti Creative, která je zvukovou alternativou k OpenGL. Aktuálně je ve verzi 1.1 a je napsána čistě pro C++ (existuje projekt pro .NET využívající knihovnu TAO).

Pomocí OpenAL se dá přehrávat i zaznamenávat zvuk, ale přímárně je určena pro přehrávání a podle toho má uzpůsobené API. Projekt obsahuje velmi rozsáhlé SDK, včetně spousty zdrojových kódů a výborné dokumentace. V SDK jsou přítomné binární soubory lib a dll podle druhu linkování, které vyžadujeme.

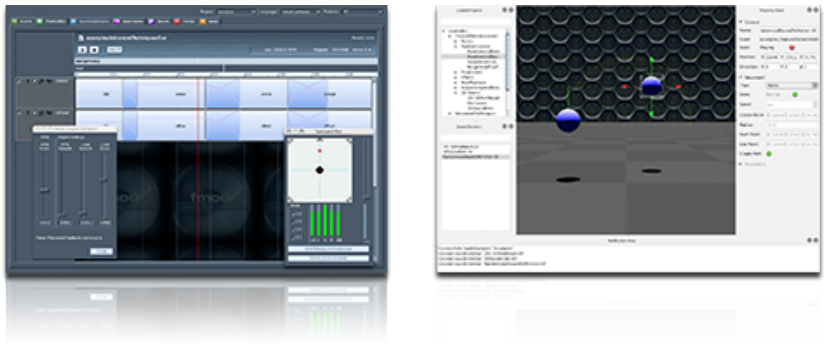
3.2.3 FMOD

FMOD [13] multiplatformní je zvuková knihovna určena především pro přehrávání (nyní ve verzi 4.10). Obsahuje vynikající dokumentaci a uživatelskou základnu včetně podpory ze strany vývojářů. Ve velké míře je využívána ve hrách, pro které je primárně vyvíjená. Mezi zajímavé vlastnosti patří:

- Podpora 3D zvuku
- Vylepšený mixážní engine
- Řadu DSP efektů
- Vícekanálový zvuk
- Podpora mnoha zvukových formátů
- FMOD Designer - Nástroj pro modelování efektů a 3D zvuku
- FMOD Sandbox - Nástorj pro nastavování 3D parametrů zvuku

FMOD má 4 licenční politiky:

- **Zdarma** - Pro nekomerční projekty
- **$1500 - $6000** - (Commercial) Pro komerční projekty
- **$750 - $3000** - (Casual) Pro komerční projekty v ceně do $20 a musí být distribuovány pouze přes internet
- **$100** - (Shareware/Hobbyist) Čistě pro PC platformu a projekt musí být vyvíjen max. jedním vývojářem



(a) FMOD Designer (b) FMOD Sandbox

Obrázek 5: Nástroje FMOD knihovny

3.2.4 DirectSound

DirectSound [14] je zvukovou komponentou knihovny DirectX [15] od společnosti Microsoft. Poskytuje rozhraní pro komunikaci mezi aplikacemi a zvukovou kartou. Než se předají zvuková data zvukové kartě, provádí DirectSound nad nimi různé funkce jako mixování kanálů nebo zvukové efekty. Od Windows Vista byla změněna zvuková architektura OS. Díky těmto změnám nebylo nadále možné využívat přímou hardwarovou akceleraci (a tím nastala ztráta výkonu). Od roku 2007 výrobci zvukových karet začali vyrábět nové zvukové karty s novými ovladači, které zachytávají volání na DirectSound API a převádí ho na své zvukové API (Creative - Creative ALchemy, Realtek - HD Audio Codecs).

Pro vývoj s využitím DirectX je k dispozici DirectX SDK, které je primárně určené pro C++. Obsahuje velké množství ukázkových zdrojových kódu a velmi dobrou dokumentaci. Pro .NET platformy existuje Managed DirectX, který ale již není nadále vyvíjen a nahradil ho XNA [16] framework. XNA framework je moderní .NET řešení DirectX, ale je primárně určen pro XBOX 360 a jeho vývoj se zastavil na verzi 9 (dokud nebude nová generace konzolí XBOX).

3.2.5 SlimDX

Jako alternativa k Managed DirectX vznikl projekt SlimDX [17]. SlimDX je open-source framework, který má za úkol převést DirectX rozhraní do .NET. Podporuje všechny aktuální verze DirectX (9, 10, 10.1, 11) a vyžadujuje minimálně .NET framework 2.0. Aktuální verze (February 2010) plně podporuje např. následující API: Math Library, DirectWrite, DirectInput a pro nás nejdůležitější DirectSound.

SlimDX SDK obsahuje velmi dobře zpracovanou dokumentaci popisující všechny přítomné API. Také obsahuje velké množství ukázkových příkladů na nejrůznější úlohy.

3.2.6 VST - Virtual Studio Technology

VST [18] je programové rozhraní od společnosti Steinberg pro komunikaci mezi hudebními syntetizéry, efekty a hudebními aplikacemi. Tato technologie pochází od společnosti Steinberg a aktuálně se nachází celkem ve 2 verzích (2.x a 3.x). VST pracuje obdobně jako DSP (Digital Signal Processing) a snaží se simulovat hardware za pomocí software.

VST se dělí na klientskou a hostovací část. Hostem jsou aplikace (většinou hudební editory), které pomocí VST rozhraní nahrávají VST klienty. Klient je buď softwarový syntetizér nebo efekt. Často mají svoje grafické rozhraní podobné nebo stejné jako u hardwarového originálu. VST mohou buď generovat zvukový signál nebo jej upravovat. Hostovací aplikace může také zřetězovat efekty a vytvářet tak komplexní efekty.

Pro vývoj VST je nutno stáhnout VST SDK [19] ze stránek Steinberg. Pro stáhnutí je potřeba si vytvořit účet. Registrace je zdarma a nemá žádné speciální požadavky. K dispozici jsou 2 verze SDK (2.x a 3.x). Tyto verze nejsou mezi sebou vzájemně kompatibilní. Celé SDK obsahuje řadu C++ tříd a rozhraní, které jsou určeny buď pro hosta nebo klienta. Existuje i několik portů do jiných jazyků (jVSTwRapper pro Javu nebo VST.NET pro .NET).

Tyto porty ovšem podporuji maximálně verzi 2.4 (která je nejvyužívanější) a pro verzi 3.x je nutné si napsát vlastní wrapper nebo projekt vyvíjet v C++. Celé SDK je pěkně zpracované a má dobrou dokumentaci i zdrojové programy.

3.2.7 ASIO - Audio Stream Input/Output

ASIO [20] je zvukový protokol/rozhraní pro ovladače zvukových karet od společnosti Steinberg. Bylo navrhnuto k poskytnutí co nejmenšího spoždění a co největšího přesnosti mezi hudebními aplikacemi a zvukovými kartami. ASIO obchází standardní cesty zpracování zvuku ve Windows a zajišťuje přímé spojení aplikace se zvukovou kartou. Takto dokáže pracovat jednoduše i s více zvkovými vstupy a výstupy nezávisle na sobě.

Aby aplikace mohla využít ASIO potenciál zvukové karty, musí tato karta ASIO podporovat. Naneštěstí tuto podporu má relativně málo zvukových karet a hlavně ji mají pouze dražší a profesionalnější karty. Proto vznikl projekt ASIO4ALL [21], který poskytuje ASIO rozhraní pro běžné karty. Jednou z „nevýhod“ ASIO rozhraní je exkluzivita používání zařízení. Pokud je jeden kanál používán jednou aplikací, nelze jej použít aplikací druhou.

3.3 Kodeky

Další z důležitých součástí pro fungování rádia jsou kodeky pro překodování nahraných dat. Dnes je jich na trhu poměrně velké množství různých kvalit. Jako standardní formát pro překodování se dnes používá formát MP3. Dalšími alternativami jsou např. OGG, APE, ACC, WMA. Každý z těchto kodeku má svoje pro a proti a některé z nich zde popíšu ty, které jsou plánovány jako klíčové pro práci.

3.3.1 LAME - Lame Ain’t an MP3 Encoder

Lame je open-source projekt, který se zabývá překodováním čistých PCM(RAW) dat do MP3 formátu. Součástí knihovny je i kód pro zpětné dekodování MP3 do PCM(RAW). Tento kód je ovšem pouze rozhraním pro knihovnu mpglib a není přímo spravován vývojáři LAME. Aktuální verze LAME je 3.98.4 (Březen 2010) a je k dispozici v několika formách (dll, exe, zdrojový kód). Dll knihovna obsahuje rozhraní pouze pro překodování PCM(RAW) dat do MP3 a zpět. Poměrně unitkátním (ale dneska již čim dal více rozšířeným) způsobem je překodování pomocí exe aplikace. Tato aplikace obsahuje veškeré možnosti knihovny LAME včetně možnosti převodu dat tam a zpět. Jedná se o konzolové rozhraní, které obsahuje poměrně rozsáhlé možnosti nastavení pomocí přepínačů a zároveň podporuje překodování přes přesměrovaný standardní vstup a výstup.

Jednou z velkých nevýhod LAME projektu je špatná dokumentace, která se víceméně dá nalézt pouze v samotných zdrojových kódech LAME. Zdrojové kódy obsahují projekty pro dll knihovnu, spustitelnou verzi LAME, zdrojové kódy mpglib, která slouží pro dekodování a pár malých ukázkových zdrojových kódů pro použití LAME. Nakonec se zde dá nalézt i seznam přepínačů a nastavení LAME, ale vše je docela nelogicky uspořádáné a je problém něco nalézt. Jednou z výhod je bezproblémová kompilace zdrojových

kódu LAME projektů. Další výhodou je kvalita překodování a hlavně rychlost, ve které je LAME prakticky na prvním místě ve srovnání s ostatními [11]. Sice v testech vychází lépe převáděč GoGo (založen na LAME 3.88), ale v testech byla použita alpha verze, které obecně nejsou plně optimalizovány. Podle diskuzí je verze 3.90+ nejrychlejší MP3 převaděč na trhu.

Jednou z vlastností MP3 formátu je nepodpora 32 bitového float-point datového typu. I když LAME podporuje datové typy float, int a long pro vstupy, ale všechny musí být převedeny do hodnot datového typu short (+/- 32768). Tímto převodem se zamozřejmě ztrácí kvalita hudebních dat.

LAME tvoří výstup ve 3 základních módech.
Constant Bitrate (CBR) - Je to výchozí mód pro převod do MP3 formátu. V tomto módu se používá stejný počet bitů k převodu pro každou část audia bez ohledu na jeho složitost překodování. Z toho vyplývá, že složitější pasáže budou mít horší kvalitu poslechu než ty lehčí. Výhodou tohoto módu je, že se výsledná velikost nemění a může být odhadnuta.
Average Bitrate (ABR) - V tomto módu převaděč má danou průměrný počet bitů, který má použít a v případě nutnosti použije více či méně bitů k převodu. Výstup je kvalitnější než u CBR, ale zachovává se možnost odhatnutí velikosti.
Variable Bitrate (VBR) - V tomto módu se vybírá výsledná kvalita místo počtů bitů k překodování (0 - 9, kde 0 je nejlepší kvalita). V tomto módu se snaží převaděč použít optimální počet bitů k převodu tak, aby dosáhl požadované kvality. Nevýhodou je nemožnost odhadu výsledné velikosti.

3.3.2 GOGO

GOGO [10] je MP3 převaděč, který je postaven na LAME (v 3.9x). Jeho hlavní předností je rychlost, která silně konkuruje samotnému LAME. GOGO podporuje pouze ABR a VBR převod.

GOGO je k dispozici jako balík zdrojových kódů a ke své kompilaci vyžaduje speciálně upravená NASM překladač (pro nejnovější verzi NASM už není třeba úprav). Samotna kompilace probíhá bez problémů a výslednými souboru jsou dll nebo lib knihovny. Nevýhodou je pouze rozhraní pro převod do MP3, ale ne zpět. Jako dokumentace slouží pouze zdrojové kódy a jejích čtení je poměrně složité, protože GOGO je čínský projekt a většina komentářů je v nečitelné formě.

3.3.3 OGG

Formát Ogg je hlavním konkurentem MP3 formátu. Vyvíjí ho nadace Xiph a jedná se open-source kontejnerový formát. Sám o sobě neslouží ke kompresi audio dat, ale k jejich formátování v souboru. Pro kompresi využívá dnes hlavně kodek Vorbis (proto mnohdy označení Ogg-Vorbis), ale může využívat i jiné. Jeho výhodou je možnost uchování metadat (např. titulků) nebo videa. Jeho formát je definován v RFC3533 a jeho MIME3.4.1 typ *application/ogg* v RFC3534.

Formát Ogg je postaven na tzv. Stránkách. Každá stránka začíná řetezcem OggS a dále pokračují dodatečné informace hlavičky. Celkově má hlavička 28 bytů a obsahuje např. typ hlavičky, verze, CRC součet, číslo stránky a další. Celkově je API Ogg formátu je relativně dobře zdokumentované a jednoduché. Samotná kompilace je bez problémů a není potřeba cokoliv provádět navíc. Jedinou nevýhodou je, že samotné zdrojové kódy neobsahují ukázkové příklady a je potřeba mít alespoň libvorbis3.3.4 zdrojové kódy, které už ukázkové příklady obsahují.

3.3.4 Vorbis

Vorbis je opět open-source nadace Xiph a jedná se o ztrátový kodek a nejčastěji bývá uložen ve formátu Ogg. Zvukově je kvalitnější než MP3, ale náročnější na převod (v obou směrech). Výsledná velikost komprimovaných souborů bývá menší než u MP3 formátu.

Jeho zdrojové kódy jsou velmi dobře zdokumentované a obsahují ukázkové příklady. Při kompilaci statické lib části nenastaly žádné problémy, ale bylo potřeba specifikovat cestu k hlavičkovému souboru ogg.h. U dynamické dll verze se opět musela nastavit cesta k ogg.h, ale při kompilaci verze 1.1.4 došlo k chybě při linkování u funkce *_analysis_output_always*, která je ve zdrojových kódech zakomentovaná, ale stále zůstala v definičním souboru.

3.3.5 FLAC - Free Lossless Audio Codec

FLAC formát je bezztrátový kodek, který využívá speciálních druhů kódování dat tak, aby nedocházelo ke ztrátám jako u ztrátových kodeků. Opět se jedná o open-source projekt nadace Xiph, který je nyní ve verzi 1.2.1 (září 2007). Vyniká svojí hudební kvalitou a díky tomu je velmi oceňovaný tam, kde je potřeba vysoká kvalita audia. Samotný FLAC je sice o něco málo méně kvalitní než jiné bezztrátové kodeky, ale zato vyniká poměrem rychlost/kvalita, kdy dokáže překódovat nahrávku v téměř stejné kvalitě za několikanásobně menší čas. Běžná výstupní velikost souboru je 60% původní velikosti.

Jednou z vlastností FLACu je možnosti obsahovat v překodovaném souboru kontejner OGG a vznikne nám tak formát OggFLAC. Rozdíl mezi standardním FLAC a OggFLAC je ten, že čistý formát je určen pouze pro jeden proud dat a maximální efektivitu přehravání. OGG kontejner je z velké části transportní a díky němu je možné spojit více proudu dat do jednoho souboru (několik audio kanálů, metadata atd.). Tato transportní vrstva dělí datový kontejner na části a díky ní je možné i data prohledávat, editovat apod.

Zdrojové kódy FLACu jsou velmi dobře zdokumentované a celkově obsahují více informací a ukázek než LAME. Jedinou nevýhodou je poněkud obtížnější kompilace zdrojových kódů a musí být provedeny různé mezikroky pro správné zkompilování alespoň základních knihoven lib a dll. Mezi nějvětší nutnosti je instalace a konfigurace NASM kompilátoru pro jazyk strojových instrukcí. Při jeho instalaci se musí nastavovat PATH proměnná systému a upravovat název souboru kompilátoru. Dále je nutné mít zdrojové kódy a knihovny Ogg knihovny, aby se provedla kompilace úspěšně.

Mezi projekty FLAC balíčku najdeme libFLAC a libFLAC++. Verze pro jazyk C++ je čistě wrapper C verze. Obě knihovny jsou ve dvou formách: statická a dynamická.

Statická forma vytvoří lib soubor, který se staticky linkuje do spustitelných souborů. Dynamická vytvoří dll knihovnu, která může být načtena dynamicky pomocí např. WinAPI LoadLibrary.

3.3.6 AAC - Advanced Audio Coding

Opět ztrátový standardizovaný formát, který byl vytvořen jako nástupce MP3. Vyniká ve středních a vyšších datových tocích v MPEG4. Tento formát není úplně jednoduchý a obsahuje různé profily, nastavení a další. Existují i různé modifikace a nejznámější z nich je AAC+ dneš již ve verzi 2.

Pro tento formát existuje řada převaděčů různých kvalit. Mezi ty základní, ale málo kvalitní patří například FAAC nebo Nero Digital a mezi ty lepší patří AAC převaděč firmy Apple. Naneštěstí podpora tohoto formátu i přes jeho kvality je špatná a většina přehravačů i dnes potřebuje nějaký dodatečný plugin. Mezi moderní přístroje, které tento formát podporují patří např. PlayStation 3, ZEN přehravače, iPod nebo PSP s aktualizovaným firmwarem.

Aktuální verze FAAC je 1.28 a je open-source. Zdrojové kódy obsahují frontend konzolovou aplikaci a knihovnu. Celá API je velice jednoduchá a základně zdokumentovaná. Chybí ukázkové příklady a překlad FAAC knihovny proběhne bez problémů.

Pro dekodování se používá FAAD2, který je open-source a nachází se ve verzi 2.7. Kompilace je opět bez problému a API je trochu víc obsáhlejší než u FAAC a dokumentace je také víceméně základní.

3.3.7 WMA

WMA kodek je součástí čistě Windows platformy a jejího Windows Media Format balíku. Je to ztrátový kodek podlehající patentu a za jeho použití se musí platit. Do verze 9 byl kvalitně na tom hůře než všechny zmíněné kodeky. Od verze 9 se však situace uplně změnila a patří dnes mezi nejvyspělejší kodeky. Jako kontejner používá formát ASF, který se také využívá pro streamování (Windows Media Streaming Services). Nevýhodou WMA je málo nastavení a obtížná implementace, ale zvukově netrpí tolika zvukovými artefakty jako konkurence a je take rychlejší. Celý Windows Media Format má dobrou dokumentaci a dá se stáhnout jako SDK.

3.4 Protokoly a standardy

Každá aplikace většinou používá nějaké protokoly nebo standardy pro svoji práci. U síťových aplikací jsou to převážně síťové protokoly, které mnohdy využívají různé další standardy nebo protokoly. Hlavními protokoly a standardy pro běh rádia jsou MIME, RTSP a Shoutcast Metadata Protocol a NGS protokol, který byl vytvořen přímo pro účely samotného rádia.

3.4.1 MIME - Multipurpose Internet Mail Extension

MIME [22] je internetový standard (RFC 2045, RFC 2046, RFC 2047, RFC 4288, RFC 4289, RFC 2049), který byl původně vyvinut pro přenost emailových zpráv. Dokáže obsahovat text i binární data. Nejvíce je dnes využíván v HTTP protokolu, ale využití má i v mnoha jiných protokolech.

Základní jednotkou MIME zpráv je párový klíč/hodnota řádek(hlavička) oddělený dvojtečkou:

klíč: hodnota

Těchto hlaviček může být libovolný počet a musí být vždy na novém řádku. Klíč musí být řetězec a hodnota může být řetězec nebo binární data. Každý protokol má svůj formát zpráv a hlaviček, ale mnohé protokoly využívají stejné hlavičky při svojí komunikaci (např. Version).

3.4.2 MMS - Microsoft Media Server

MMS [33] byl hlavním protokolem pro síťové streamování dat ve Windows Media Services od společnosti Microsoft. MMS podporuje jak TCP, tak i UDP vrstvy a standardním portem pro přenos je 1755. Microsoft ho označil za zastaralý a přešel na RTSP protokol. Od verze Windows Server 2008 již není MMS podporováno.

3.4.3 RTSP - RealTime Streaming Protocol

RTSP [32] je síťový protokol vytvořený pro přenos streamovaných dat. Je to nejrobustnější a nejuniverzálnější protokol, který lze dnes najít. Jeho specifikaci lze nalézt pod dokumentem RFC2326. Samotný protokol je velice podobný jako HTTP, ale přidává několik typů požadavků navíc a je plně stavový. Při přenosu se posílají session hlavičky a není tedy potřeba trvalého TCP/UDP připojení. Výchozím portem pro RTSP protokol je 554. Jak bylo řečeno, RTSP přidává k protokolu HTTP několik požadavků, které využívá pro svoji práci. Zde je jejich výčet těch důležitých:

OPTIONS požadavek slouží ke zjištění informací od serveru. Při požadavku server vrací podporované typy, požadavky atd.

DESCRIBE požadavek je nejdůležitějším požadavkem v komunikaci. Tento požadavek získá od serveru popis média, který je požadován. V tomto popisu jsou většinou hlavičky, které určují např. název serveru, název streamu a nastavení streamu (většinou pomocí SDP protokolu).

SETUP požadavek slouží k nastavení přenosového mechanismu pro stream. Většinou je zde využíván protokol RTP.

PLAY požadavek řekne serveru, aby začal posílat data. Tento požadavek musí být vždy až po požadavku SETUP.

TEARDOWN požadavek řekne serveru, aby ukončil posílání dat a uvolnil obsazené zdroje apod.

```
DESCRIBE rtsp://**********.cz/radio?WMContentBitrate=135000 RTSP/1.0
User-Agent: WMPlayer/11.0.6001.7000 guid/3300AD50-2C39-46C0-AE0A-CDA4877498FB
Accept: application/sdp
Accept-Charset: UTF-8, *;q=0.1
X-Accept-Authentication: Negotiate, NTLM, Digest, Basic
Accept-Language: cs-CZ, *;q=0.1
CSeq: 1
Supported: com.microsoft.wm.srvppair, com.microsoft.wm.sswitch, com.microsoft.wm.eosmsg, com.
    microsoft.wm.predstrm, com.microsoft.wm.startupprofile
```

Výpis 1: Ukázka požadavku na stream

```
RTSP/1.0 200 OK
Content-Type: application/sdp
Vary: Accept
X-Playlist-Gen-Id: 2710239
X-Broadcast-Id: 2434913
Content-Length: 7520
Date: Tue, 10 Feb 2009 19:32:23 GMT
CSeq: 1
Server: WMServer/9.1.1.5000
Supported: com.microsoft.wm.srvppair, com.microsoft.wm.sswitch, com.microsoft.wm.eosmsg, com.
    microsoft.wm.fastcache, com.microsoft.wm.packetpairssrc, com.microsoft.wm.startupprofile
Last-Modified: Sat, 30 Dec 1899 00:00:00 GMT
Cache-Control: x-wms-stream-type="broadcast", no-cache, no-user-cache, private

v=0
o=- 200901192250200035 200901192250200035 IN IP4 127.0.0.1
s=<No Title>
c=IN IP4 0.0.0.0
b=AS:129
a=maxps:5976
t=0 0
a=control:rtsp://netshow4.play.cz/kissmoravaov128/
a=etag:{68643A14-D4AD-ADB6-9D8D-7E3A5E03BA11}
a=range:npt=1.579-1.579
a=type:broadcast
a=recvonly
m=audio 0 RTP/AVP 96
b=AS:129
b=X-AV:129
b=RS:0
b=RR:0
a=rtpmap:96 x-asf-pf/1000
a=control:audio
a=stream:1
m=application 0 RTP/AVP 96
b=RS:0
b=RR:0
a=rtpmap:96 x-wms-rtx/1000
a=control:rtx
a=stream:65536
```

Výpis 2: Ukázka odpovědi na stream (včetně SDP definic)

3.4.4 SDP protokol

SDP [34] je předpis, pro popis streamovaných médií při inicializaci spojení. Je to textových formát založený na MIME formátu zpráv a je definován v RFC4566. Samotný protokol nepřenáší žádná data a je navžen tak, aby byl dobře rozšířitelný pro nové formáty. Mezi základní definiční hlavičky patří např. version, session name, uri, email, phone, media title a další. Ukázku SDP lze nalézt v druhém bloku výpisu 2.

3.4.5 RTP protokol

RTP [35] je transportním protokolem pro přenos audia/videa. RTP je párový protokol a pracuje společně s RTCP protokolem, kde RTP slouží pro samotný přenos dat a RTCP pro přenost kontrolních informací. RTP běží jak nad TCP, tak i nad UDP. Samotný RTP protokol má několik RFC dokumentů, kde každý z nich popisuje přenost jiného druhu média (audio, video, text atd.).

3.4.6 Shoutcast Metadata Protocol

Shoutcast Metadata Protocol [23][24][25] protokol byl vytvořen firmou NULLSOFT pro Shoutcast Radio server. Tento protokol využíval primárně přehravač Winamp a samotný protokol není open-source. Protokol byl pomocí reverzního inženýrství zdokumentován a začaly ho využívat i další aplikace. Protokol je postavený na MIME zprávách a je poměrně jednoduchý.

Samotný protokol neobsahuje velké množství hlaviček a tim pádem i nastavení. Všechny jeho hlavičky začínají prefixem *icy-* a za pomlčkou se nachází identifikátor. Seznam nejpoužívanějších hlaviček:

- **icy-name** - Název stanice
- **icy-br** - Bitový tok (Bitrate)
- **icy-metadata** - Počet bytů mezi bloky metadat
- **icy-url** - URL stanice

Nejdůležitější hlavičkou je *icy-metadata*. Tato hlavička určuje, kolik bytů dat je mezi bloky metadat. Kdybychom měli např. *icy-metadata: 8192*, tak se blok metadat bude nacházet každý 8192 bytů.

Obrázek 6: Formát metadat v produ dat

Na obrázku lze vidět, jak asi vypadá blok metadat v toku dat. Malý zelený proužek určuje, jak velký bude blok metadat. Tato velikost je určena pouze jedním bytem, u kterého se jeho hodnota násobí 16. Jelikož byte má maximální rozsah $\langle 0; 255 \rangle$, tak maximální velikost bloku metadat může být 4080 bytů.

Blok metadat může obsahovat libovolná data, ale nejčastěji obsahuje řetězcová data (např. *StreamTitle=Ńázev skladby´*. Z předchozího odstavce je jasné, že blok musí mít velikost dělitelnou 16, ale řetězce většinou tuto velikost nemají, proto se zbyvající data vyplňují \0, aby se blok zarovnal.

```
GET /stream/1065 HTTP/1.1
Icy-MetaData: 1
User-Agent: BASS/2.4
Host: **********.com
Cache-Control: no-cache
```

Výpis 3: Ukázka požadavku na stream

```
ICY 200 OK
icy-metadata: 0
icy-br: 128
icy-name: Ultimate Radio
```

Výpis 4: Ukázka odpovědi na stream

3.5 Protokol NGS

Protokol NGS byl navržen pro komunikaci mezi klienty a serverem. Vychází stejně jako předchozí protokoly z MIME zpráv. Celá komunikace funguje tak, že klient zašle konfigurační zprávu, která obsahuje přihlašovací údaje a nastavení pro danou stanici. Server po autorizaci vytvoří instanci rádia podle konfigurace a kdyby autorizace selhala, tak se spojení ukončí.
Hlavičky NGS protokolu:

1. **NGSEncoder** - Kodek použitý pro kompresi dat
2. **NGSBitrate** - Bitový tok dat
3. **NGSLogin** - Přihlašovací jméno
4. **NGSPassword** - Přihlašovací heslo (v hašovaném tvaru)

3.6 Ostatní technologie

V této kapitole si probereme technologie, které nejsou kriticky důležité, ale dají se využít a mohou zpřijemnit používání rádia. Nebudou popsány nějak podrobně a pro dodatečné informace je potřeba použít jiných zdrojů.

3.6.1 ID3 tagy

ID3 [26][27] tagy jsou tagovací systém pro hudební soubory. Primárně byly vytvořeny pro hudební formát MP3, ale později je začly využívat i jiné formáty. ID3 je vlastně datový kontejner, který je umístěn většinou na začátku (může být ale i na konci) souboru. Obsahuje většinou informace o hudební nahrávce (např. autora, název skladby, žánr, obrázek a spoustu dalších). Jak lze z výčtu informací vyčíst, tak data můžou být jak řetězcová, tak i binární. Většina dnešních přehravačů dokáží správně tyto data přečíst a zobrazit při přehrávání.

Nyní jsou používány 2 verze ID3 tagů. ID3v1 a ID3v2.x (nejnovější je 2.4, ale kvůli jistým omezením je spíše využívána verze 2.3). Základní jednotkou tagů je Frame, který obsahuje požadované informace a může mít až 16 MB. Maximální velikost tagů dohromady může být až 256 MB.

Existuje velké množství implementací pro různé jazyky. Některé jsou více zdařilé a některé méně. Pro C# doporučuju TagLib [28] knihovnu, která vychází z C++ verze a je poměrně snadná na naučení a je dost robustní pro komplexní práci.

3.6.2 Playlisty

Playlisty slouží pro vytváření seznamů skladeb, které se mají přehrát. V našém případě nebudeme tvořit seznam skladeb, ale seznam rádio stanic, které chceme přehrát. Uživatel sice může zadat pro poslech přímou adresu rádia, ale je to zdlouhavý a namáhavý způsob (a pro uživatele otravný). Navíc v situaci, kdy máme více serverů, které zrcadlí datový proud rádia, tak by byl uživatel nucen zkoušet více adres, než by našel tu, která by ho přijala (např. kvůli vytížení nebo omezení počtů posluchačů).

Pro usnadnění těchto úkonů můžeme využít playlistů, kdy stačí, aby ho uživatel spustil a pokud má nainstalován přehravač, který dokáže tento playlist přehrát, tak se mu automaticky začnou zkoušet nastavené adresy rádia, dokud se nenajde volná pozice. Dále playlisty mohou obsahovat určitá metadata (např. název stanice), které mohou přehravače využít. Základními formáty playlistů jsou *pls*, *m3u* a *asx*.

PLS [29] je nejjednodušším formátem ze všech 3 zmíněných formátů. Je velmi podobný INI formátu a využívá ho drtivá většina dnešních přehravačů. Playlist může vypadat asi takto:

```
[ playlist ]

File1=http://streamexample.com:80
Title1=My Favorite Online Radio
Length1=-1

File2=http://example.com/song.mp3
Title2=Remote MP3
Length2=286

File3=/home/myaccount/album.flac
Title3=Local album
Length3=3487
```

```
NumberOfEntries=3

Version=2
```

Výpis 5: Ukázka pls formátu

Na začátku playlistu musí být hlavička *[playlist]*, která indikuje, že se jedná playlist. Dále následují jednotlivé položky playlistu, u kterých je zadána cesta, titulek a délka skladby. Pokud je délka -1, znamená to, že nemá délku definovanou (většinou skladba nemá konec - většinou u rádií). Na konci je počet počet položek a verze, která nyní podporuje pouze hodnotu 2.

M3U [30] je méně využívaný formát od společnosti NULLSOFT. Původně byl navržen čistě pro přehravač Winamp, ale dnes je již podporován celou řadou přehravačů. Základem formátu je #, který označuje počátek direktivy. M3U má 2 hlavní direktivy *#EXTM3U* a *EXTINF*. Direktiva *#EXTM3U* musí bát na začátku souboru a opět určuje, že se jedná o playlist. *EXTINF* obsahuje informace o délce skladby, autorovi a názvu skladby (délka, autor - název skladby) a pod touto direktivou je cesta k souboru. Ukázka playlistu:

```
#EXTM3U

#EXTINF:123,Sample Artist – Sample title
C:\Documents and Settings\I\My Music\Sample.mp3

#EXTINF:321,Example Artist – Example title
C:\Documents and Settings\I\My Music\Greatest Hits\Example.ogg
```

Výpis 6: Ukázka m3u formátu

ASX [31] (Advanced Stream Redirector) je playlist založený na XML technologii a využívá ho především Windows Media Player. Jelikož je XML formát volný, tak může prakticky obsahovat libovolné textové informace. Ukázka asx formátu:

```
<asx version="3.0">
  <title>Example.com Live Stream</title>

  <entry>
    <title>Short Announcement to Play Before Main Stream</title>
    <ref href="http :// example.com/announcement.wma" />
    <param name="aParameterName" value="aParameterValue" />
  </entry>

  <entry>
    <title>Example radio</title>
    <ref href="http :// example.com:8080" />
    <author>Example.com</author>
    <copyright>©2005 Example.com</copyright>
  </entry>
</asx>
```

Výpis 7: Ukázka asx formátu

Jako kořenový element je ASX element, který by měl mít specifikovanou verzi. Následuje element title, který určuje název celého playlistu. Poté jsou jednotlivé položky playlistu, které mohou mít autora, název, odkaz, různé parametry a další libovolné elementy a atributy.

4 NGS Radio Server

4.1 Základní popis

NGS Radio Server je vyvíjená aplikace spolu s touto prací. Zkratka NGS označuje **Next Generation Streaming**, která podle mě vystihuje směr vývoje projektu a jeho myšlenku. Cílem projektu není vytvořit technologicky inovativní dokonalé dílo (v jednom člověku toto ani nelze), ale aplikaci, která bude mít tyto rysy:

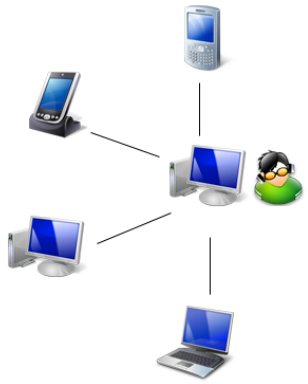
- Jednoduchý pro běžného uživatele
- Dostatečně komplexní pro náročnější uživatele
- Vzhledově přívětivý
- Nepřeplácaný
- Snadný na pochopení a ovládání
- Určeny pro všechny: DJ's, rádiové stanice, víkendového amatéra nebo profesionála
- Softwarově nezávislý s co největší možností používat libovolné aplikace třetích stran

Projekt je vyvíjen ve Visual Studiu 2008 na platformě Windows Vista/7 s využitím .NET 3.5 SP1 frameworku. Použitými jazyky jsou C#, C++ (případně C++/CLI) pro klientskou i serverovou část. Dále v serverové části je použita technologie ASP.NET a databáze SQL Compact 3.5 SP1 včetně dodazovacího jazyka LINQ a technologie WCF.

4.2 Základní koncepty

Aplikace má čtyři základní koncepty na kterých bude postavena výsledná aplikace.

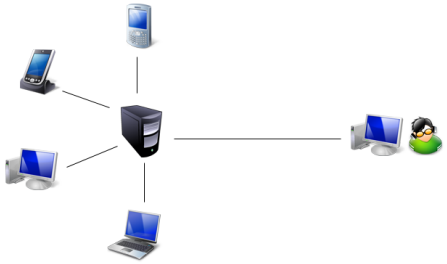
4.2.1 Koncept 1. - Client only



Obrázek 7: Koncept 1. - Client only

První koncept je postaven na principu, kdy klientský stroj je zároveň i vysílací server. Tento návrh je vhodný převážně pro uživatele, kteří chtějí vysílat omezenému okruhu lidí (většinou ne více než 10 lidem). Samozřejmě mohou být výjimky pokud to dovolí výpočetní kapacita a datová linka klienta. Na klientském stroji bude nainstalována speciálni verze rádiového softwaru, která bude mít integrovanou základní část serverové části.

4.2.2 Koncept 2. - Server - Client

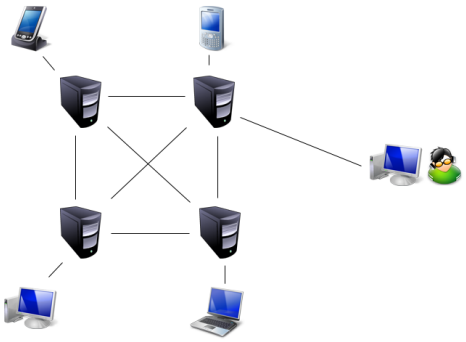


Obrázek 8: Koncept 2. - Server - Client

Druhý koncept je stěžejním návrhem celého projektu. Jeho myšlenka spočívá v rozdělení rádiového softwaru na 2 samostatně fungující jednotky. Klientská jednotka má za úlohu

zachytávat na klientovi audio data, zkomprimovat je a odeslat na serverovou část, která tato data přepošle mezi připojené posluchače. Tento návrh je vhodný pro větší střední počet připojených uživatelů (desítky až stovky).

4.2.3 Koncept 3. - Distributed Servers - Client



Obrázek 9: Koncept 3. - Distributed Servers - Client

Třetím konceptem je distribuovaný návrh. Tento návrh je založen na Server - Klient konceptu, ale je rozšířen o možnost distribuovaného rozložení zátěže. V zásadě je princip fungování takový, že klient odešle data na jeden ze serverů, ten je následně přepošle svým sousedům. Připojení posluchači jsou následně přesměrováni na nejvhodnější server, od kterého budou přijímat data.

4.2.4 Koncept 4. - Streaming Farms

Posledním konceptem jsou streamovací farmy. Vychází z návrhu Distribuovaných serverů - klientů, kdy se provede ještě jemnější rozdělení jednotlivých úloh s účelem rozdělení zátěže na více počítačů. Z klienta se odešlou audio data v co nejlepší kvalitě na překódovací server, který daná data překóduje do různých formátů a kvalit a následně je přepošle na streamovací servery, které si je úloží do zásobníků. Potom master server obsluhuje klienty, kteří se připojují a ty přepojí na streamovací servery. Tento návrh je vhodný pro velký počet klientů (stovky až tisíce).

4.3 Princip zaznamenávaní dat

Jedním ze základním problémů při tvorbě rádia byl způsob záznamu zvuku. Jelikož celá záznamová část je postavena na principu aplikace třetí strany, tak nemáme možnost přímo získávat data například přehravače Winamp. Tento problém se dá vyřešit 3 způsoby: Stereo mixem, Virtualními kabely/Virtuální zvuková karta.

4.3.1 Stereo mix

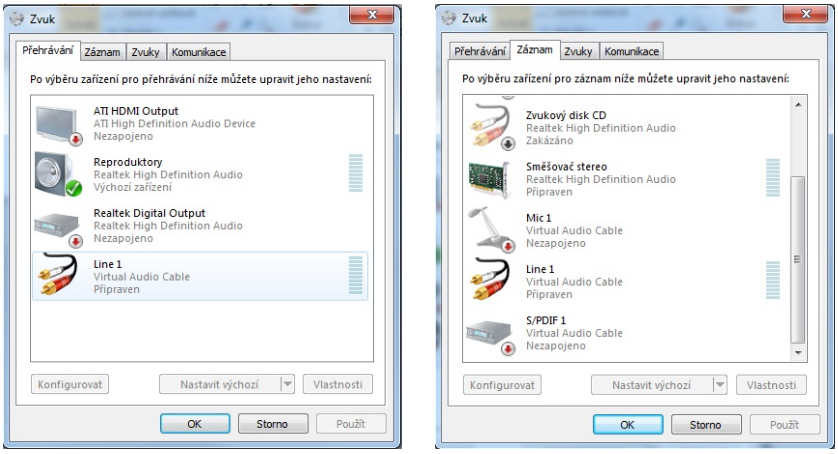
Stereo mix (na některých kartách také jako What U Hear) je technologie zvukových karet, která vytváří tzv. Loopback (zpětná vazba/smyčka). V podstatě jde o to, že veškerá data, která projdou přes zvukovou kartu na výstup se automaticky okopírují a přepošlou do Stereo mixu, který slouží jako vstup, proto ho můžeme použít jako záznamové zařízení. Tento způsob má 2 hlavní nevýhody:

- Zvuková karta MUSÍ podporovat Stereo mix
- Nelze rozlišit zdroj dat

Naštěstí dneska prakticky všechny zvukové karty (i integrované v základních deskách) podporují Stereo mix a není tedy problém. Největší nevýhodou tohoto řešení je nemožnost rozlišit zdroj dat. Tudíž, pokud máme 2 aplikace, které budou navzájem přehrávat 2 různé skladby, tak ve Stereo mixu budeme mít obě skladby zmixované jako jednu. Tudíž cokoliv, co se nám promítne do přehravání se promítne do vysílání. Tyto nevýhody částečně vyvažuje poměrně snadné použití v aplikaci.

4.3.2 VAC - Virtual Audio Cable / Virtuální zvuková karta

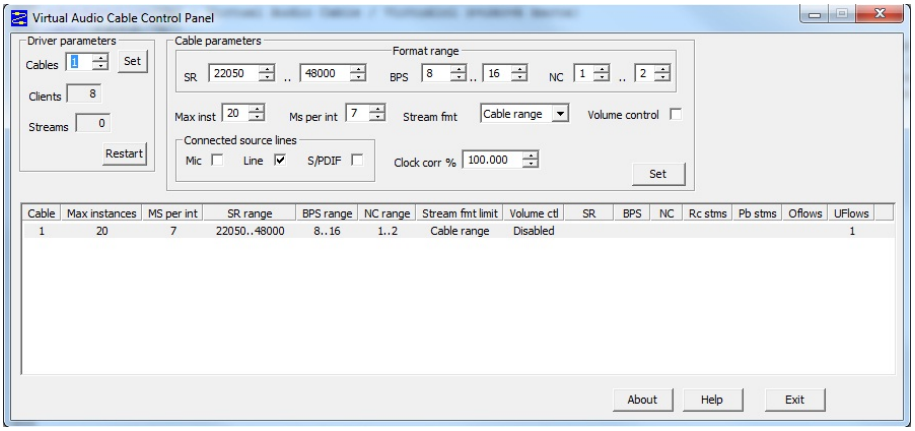
Druhým a již lepším řešení je použití Virtuální kabelů[7]. Pomocí VAC lze přenášet audio data mezi aplikacemi nebo zařízeními. Jedná se v podstatě o ovladač virtuální zvukové karty, který vytvoří vstupní/výstupní zařízení (kabely), které se pro ostatní aplikace jeví jako fyzické zařízení a při jejich použití lze přesměrovávat tok dat z/do požadovaných zařízení. Samozřejmostí je přesměrování dat do fyzické zvukové karty a umožnit jejich poslech na zvukovém systému. Momentálně je ovladač ve verzi 4.10 za cenu 30 amerických dolarů. Problém s instalací a funkcí ovladaců by neměl nastat ani na Windows 7. Jediný případ, kdy může dojít k potížím jsou X64 systémy, kde jsou vyžadovány digitálně podepsané ovladače a pro správný chod je potřeba mít aktivovaný testovací režim. Po instalaci ovladače se objeví kabely v nastavení zvuku.



(a) Výstupní zařízení (b) Vstupní zařízení

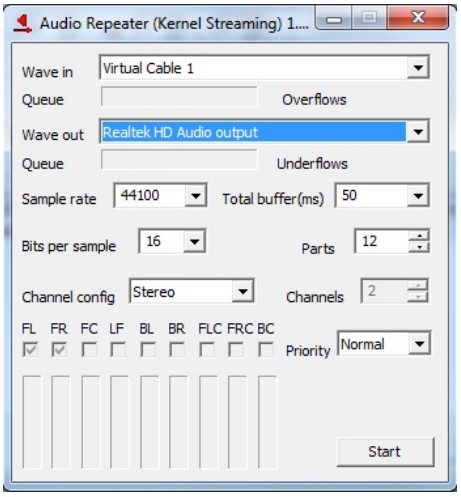
Obrázek 10: Nastavení zvuku - Windows 7

Základním ovládacím prvkem VAC je kontrolní panel. V něm se dájí přidávat a odebírat virtuální kabely. Zároveň obsahuje poměrně dost nastavení pro jednotlivé kabely. Mezi tyto nastavení patří např. frekvenční rozsah, počet bitů na vzorek nebo počet kanálů.



Obrázek 11: VAC - Kontrolní panel

Samotné virtuální kabely nepřeposílají data na standardní výstup a nejde tedy slyšet to, co se přehrává. Na tuhle situaci autor myslel a vytvoři program **Audio Repeater**. Tato aplikace slouží k přeposílání jednoho zdroje dat do druhého. Dají se nastavit např. jaké kanály se mají přeposílat, v jaké frekvenci nebo jaký se má použít počet bitů na vzorek.



Obrázek 12: VAC - Audio Repeater

4.4 Knihovny

V celém projektu byla snaha ho vést jako návrhový vzor Model-View-Controler (MVC). Nejvíce lze tento model vidět v serverové části, kde byla snaha o co nejlepší rozdělení a univerzálnost.

Celý solution (projekt) rádia obsahuje řadu podprojektů, z nichž jsou většina dll knihovny. Mezi tyto knihovny patří **NGSCore**, **NGSServer**, **NGSClient**, **BassEngine**, **Lame_encdec**, **LameEncoder**. Poté jsou v přítomny ještě 2 spustitelné soubory **NGSServerGUI** a **NGSClientGUI**. Nyní si je popíšem včetně problémů při vývoji:

4.4.1 Jádro

NGSCore je nejdůležitější knihovnovnou v celém projektu a obsahuje veškeré sdílené třídy, události, delegáty, výjimky apod. jak pro serverovou, tak i pro klientskou část, a proto jí využívají prakticky všechny ostatní knihovny. Zde popíšu jen ty nejvýznamnější.

MIME parser je první a jednou z nejdůležitějších klekcí tříd v celém jádru. O MIME zprávách je již napsáno v kapitole 3.4.1 a pro ně vytvořená tato implementace. Obsahuje 4 třídy: HeadersException, Headers, Interfaces, MIMEMessage.

Interfaces obsahuje metody a property, které musí implementovat každá hlavička, která má být zpracovávána parserem. Tyto hlavičky jsou pak naimplementovány v *Headers*.

MIMEMessage je hlavním parserem pro MIME zprávy. Celá implementace funguje tak, že se z hlaviček, které implementují rozhraní IHeader vytvoří tabulka, která se pak použije pro pro samotné parsování zprávy, kdy se postupně prochází hlavičky a volá se vnitřní parser hlaviček, které již ví, jak se správně samy sebe zpracovat. Opačným procesem je renderování, kdy se z hlaviček vyrenderuje celá MIME zpráva jako řetězec. Zde je možnost rozšířit renderování do XML, ale momentálně tato vlastnost není implementována.

CircularQueueBuffer je speciální typ fronty, která byla vytvořena pro účely rádia. Z názvu již vyplývá, že speciální vlastností je kruhovost. Aby byl princip bufferu správný,

tak musí mít pevnou velikost, která se zvolí na začátku (kapacita). Poté probíhá vkládání a výběr dat z buferru normálním způsobem, jako v normální implementaci fronty. Rozdíl je ve vnitřní implementaci, kdy jsou přítomny 2 indikátory: BEGIN a END. BEGIN označuje počátek dat a END konec dat. Když se vloží data, posune se indikátor END dále od BEGIN, naopak při výběru se posune indikátor BEGIN blíže k END. A nyní hlavní rozdíl: posouvají se pouze oba indikátory neustále jakoby dokola. Tím nám vznikne „nekonečný" buffer, kdy se stará data přepíši novými a celý buffer se vlastně točí. Další vlastností je, že pokud náhodou narazí END na BEGIN (třeba při nedostatečném vybírání), tak END tlačí BEGIN jakoby před sebou.

Další třídou, kterou CQB využívá pro účely rádia je *QueueListEx*, který je vlasté kombinací zásobníku a fronty, kdy lze provádět vkládání jak zezadu, tak i zepředu a stejně tak i výběr. CQB tento list využívá pro uchovávání velikostí vložených bloků, aby se při výběrů vybral stejný počet dat, jako při vkládání. Tím se zajistí posílání přesného počtu dat pro bloky o délce 1 sekundy.

EncodersManager je třída, která se stará o nahrávání pluginů pro překódování dat (např. do MP3). Vyhledávání pluginů se provádí tak, že se nejdříve vyhledávají soubory ve formátu *ngs_*.dll*, poté se zkontroluje, jestli obsahují implementaci rozhraní *IEncodersInterface* a poté se vytvoří jejich instance. Každá implementace obsahuje speciální property ShortName, pomocí které se pak dá požadovaný encodér získat (identifikovat) a použít.

BaseSenderBuffer je bázová třída, kterou dědí třídy SourceRecorder (viz. 4.4.5) a RadioChannel (viz. 4.4.7). Tyto dvě zmíněné třídy do této vkládají hotová a zpracovaná data, která již jsou připravená k odeslání. Pro uložení se využívá CircularQueueBuffer a jako časovač je použit Timer, který co 1 sekundu vyvolá událost DataReady, která značí, že je dostatek dat a pravý čas k jejich odeslání.

4.4.2 BassEngine

Tato knihovna je hlavní knihovnou sloužící pro záznam audio dat a využívá knihovny Bass a jejího wrapperu Bass.NET. Základem je implementace rozhraní *IRecordEngine*, které obsahuje potřebné metody a property pro manipulaci. Na začátku se provede inicializace a na konci uvolnění prostředků. Dále se zjišťují nahrávací zařízení a vstupy, které jsou k dispozici v systému. Poté už se provádí samotné spouštění a vypínání nahrávání. Důležitým prvkem je událost *OnRecoderBufferFilled*, která se musí zavolat, je li nahrán dostatečný počet dat a tyto data se následně předají k dalšímu zpracování (k převodu do jiného formátu). Veškerá vnitřní implemenace je silně závislá na tom, jaká technologie je použitá k nahrávání a podle toho se odvíjí. Momentálně je nahrávací engine nastaven napevno a není uděláno dynamické načítaní jako u překódovacích knihoven.

4.4.3 Lame_encdec

Lame_encdec je hlavní knihovnou pro převod surových audio dat do MP3 formátu a zpět napsána v C++. Vývoj této knihovny se ovšem ukázal jako jeden z největších problémů vůbec během celého vývoje rádia.

Prvotním návrhem bylo mít jednu instanci (při velkém vytížení třeba i víc) encodéru/decodéru pro více rádií, kdy by např. jeden encodér MP3 převáděl data pro x rádií. Toto se ovšem ukázalo jako nemožné. Pro práci s MP3 formátem byla použita knihovna LAME (viz. kapitola 3.3.1), která obsahuje metody pro převod tam i zpět a využívají jí prakticky každá jiná implementace MP3.

Prvním problémem je, že všechny encodéry nechávají na začátku překódovaného bloku malé tichého místa (nějakých 0,23 sekundy) a na konci se provádí zarovnání prázdnými byty. Tím nám při postupném převodu 1 sekundových bloků vznikájí malé mezery a výsledný audio stream není plynulý na poslech.

Druhým problémem je velmi špatná podpora vícevláknového zpracování LAME knihovny. Samotná knihovna obsahuje velké množství statických dat, které zabraňují správnému vícevláknovému použití. Problém nastal například tehdy, když se pro jedno rádio vytvořilo více kanálů (např. 128 a 32), kdy se nejdříve inicializoval 128 bit kanál a poté 32 bit, tak stejně 128 bit kanál překódoval jako 32 bit, protože ve statických proměnných se uchovává pouze poslední nastavení. Tím jsem byl nucen vymyslet způsob, jak tento problém obejít (viz. následující kapitola).

4.4.4 LameEncoder

Tato knihovna je v podstatě velký wrapper pro manipulaci s C++ verzí knihovny pro převod dat MP3. Samotný podprojekt obsahuje 3 třídy: LameWrapper, LibraryManager, MPEGLame.

LibraryManager je nejdůležitější třídou pro tuto knihovnu. Jak bylo v předchozí kapitole zmíněné, tak LAME knihovna je velmi špatně napsaná pro použití ve vícevláknových aplikacích. Jelikož původní návrh, kdy jedna instance bude zpracovávat více zdrojů naráz nemůže fungovat, a proto se musí použít návrh, kdy každá intance encodéru bude obsluhovat jeden kanál. Ovšem zde je jeden významný problém. Abychom mohli mít více instancí jedné knihovny opravdu s oddělenými zdroji, musíme tuto knihovnu nahrát dynamicky pokaždé, když tvoříme novou instanci knihovny. Toto dynamické nahrávání se provádí pomocí WinAPI LoadLibrary a GetProcAddress, ale i toto nahrávání má jisté omezení. Pokud nahrajeme knihovnu do paměťového prostoru procesu pomocí LoadLibrary, získame na tuto knihovnu handle. Pokud ovšem opět pomocí LoadLibrary nahrajeme knihovnu se stejným názvem a která už byla jednou nahrána, tak získame handle na tu již přítomnou (nahranou) v paměti. Tím pádem máme jednu kopii knihovny v paměti a jediné, čeho dosáhneme vícenásobným použitím LoadLibrary je zvýšení vnitřního čítače na knihovnu. Tato vlastnost byla zavedena na 16 bitových systémech, kdy byl ještě nedostatek paměti a přetrvala dodnes.

Abychom toto omezení obešli, musíme pokážde nahrávat stejnou knihovnu, ale s jiným názvem. O toto se stará LibraryManager, který ma metodu *LoadDll* a ta při zavolání zkopíruje původní LAME knihovnu do Temp adresáře s vygenerovaným jménem a tento překopírovaý soubor se nahraje. Tím se vlastně obejde omezení systému, který nekontroluje vnitřní hlavičky knihoven, ale jen název knihovny a tím nám dovolí jednu knihovnu nahrávat dle naších potřeb. Při uvolnění knihovny se tyto dočasné soubory smažou.

LameWrapper je samotný wrapper, který při inicializaci zavolá LoadDll metodu v LibraryManager a získá handle na nově nahranou knihovnu. Pomocí tohoto handle získá pomocí GetProcAddress z knihovny adresy metod, které se poté převedou Marshalingem na delegáty, které mohou být použity jako klasické funkce.

MPEGLame je třída, která implementuje rozhraní IEncodersInterface a zajišťuje komunikaci mezi hlavní aplikací a převáděcí knihovnou. Všechny encodéry, které mají tvar *ngs_*.dll* a implementují rozhraní IEncodersInterface jsou dynamicky nahrány a je možné je použít pro překodování.

4.4.5 NGSClient

Tato knihovna slouží k řízení nahrávání a následnému odesílání dat zaznamenaných a zpracovaných dat na server. Všechny tyto činnosti obsluhují 2 třídy: RadioClient a SourceRecorder.

RadioClient obsahuje 3 metody: ActivateSending, DeactivateSending a SendData. Pro posílání dat je využit nejnižší možná třída Socket. Pomocí ní se vytvoří TCP instance, která se pokusí připojit na server. Po připojení se odešle MIME zpráva, která obsahuje hlavičky NGSLogin, NGSPassword, NGSEncoder, NGSBitrate. NGSLogin obsahuje přihlašovací jméno, NGSPassword přihlašovací heslo v hašovaném SHA256 tvaru. NGSEncoder obsahuje informace o použitém encodéru (jeho ShortName) a NGSBitrate datový tok překódovaných dat. Na metodu SendData je napojena událost DataReady ze třídy BaseSenderBuffer, která po vyvolání odešle data na připojený socket.

SourceRecorder dědí ze třídy BaseSenderBuffer a stará se o samotné zaznamenávání dat a v pravý čas o jejich odeslání. Nejdůležitějším prvekm této třídy je událost OnRecordBufferFilled, která je zavolána tehdy, když se nahraje dostatek dat. V této události se poté provede překodování zaznamenaných dat do zvoleného formátu a jejich ïaskladněnív bázové třídě BaseSenderBuffer, která je pak při dostatečném počtu odešle (ve výchozím stavu je zapotřebí mít minimálně v bufferu 3 sekundy dat, než se začnou postupně bloky odesílat, ale toto nastavení je možné změnit).

4.4.6 NGSClientGUI

Tento program slouží jako uživatelské rozhraní pro klientské knihovny. Celé GUI je postaveno na technologii WPF (Windows Presentation Foundation). Tuto technologii jsem zvolil z důvodu velmi jednoduché změny vzhledu díky šablonám a dostatečné škálovatelnosti, která je mnohem větší než u Windows Forms. Celé rozhraní je velice základní a v nynější podobě poměrně nevzhledné a proto byla zvolena technologie WPF, kdy specialsta na uživatelské rozhraní a grafik mohou poměrně jednoduše nakreslit vzhled v Expression Blend a následně tento vzhled aplikovat do programu.

Celé rozhraní využívá klasický systém událostí na komponenty a pomocí nich je řízeno celé chování aplikace. Na počátku aplikace se načte základní nastavení z xml souboru pomocí třídy *NGSConfigManager*. Tato třída využíva strukturu NGSConfig, která obsahuje proměnné, které chceme používat napříč celou klientskou aplikací. Mezi tyto proměnné patří např. ServerIP, Port, Login a Password. Login a Password jsem se rozhodl neukládat,

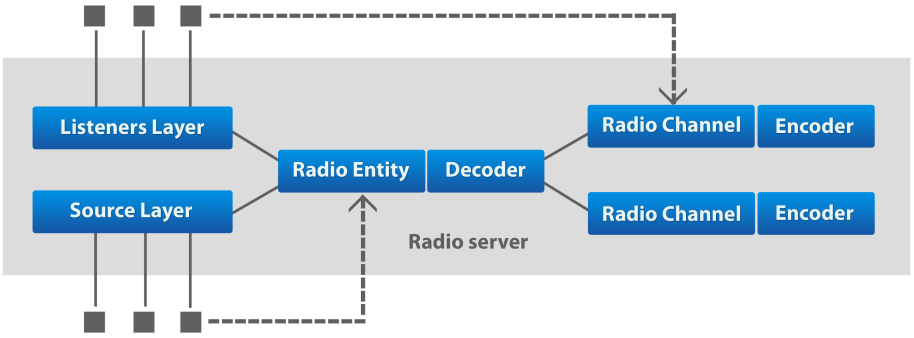
tak sem vytvořil speciální atribut NoSave, který lze na tyto proměnné aplikovat a jím označené budou při načítání a ukládání ignorovány. Ukládání a načítání probíhá pomocí reflexe, kdy název proměnné ve strukture NGSConfig je stejný jako název elementu v xml. Pokud konfigurační soubor neexistuje, tak se při prvním ukončení aplikace vytvoří.

Nejzájímavějším prvkem implementace GUI je delegát *BroadcastStopper*. Tento delegát bylo nutné zavést z důvodu vícevláknového zpracování, kdy vlákno, které neobsluhuje GUI přímo nemůže měnit vlastnosti GUI komponent. Proto byl zaveden BroadcastStopper, který se zavolá jako událost při ukončení spojení se servem. Ten pomocí Dispatcheru (objekt pro práci ve vícevláknovém prostředí) asynchronně invokuje metodu, která provádí veškeré náležitosti spojené s vypnutím vysílání a nahrávání.

Uživatelská příručka pro práci s klientem je v Dodatku A (kapitola 8.1.1).

4.4.7 NGSServer

NGSServer knihovna je nejdůležitější částí celého projektu. Obstarává komunikaci mezi klienty, na kterých probíhá nahrávání audio dat a posluchači. Celou architekturu lze vidět na obrázku.



Obrázek 13: Schéma architektury serveru

Výchozí třídou celého serveru je **RadioServer**. Tato třída obsahuje kolekci již aktivních rádii a 2 spojovací vrstvy: *Source Connection Layer* a *Listeners Connection Layer*, z nichž každá obsluhuje příchozí požadavky na server pro určitou stranu.

Source Connection Layer obsluhuje příchozí požadavku od klientských stanic, na kterých probíhá záznam dat a následné odesílání na server. Tato vrstva běží na portu 3443 nad protokolem TCP za použití nejnižší komunikační třídy .NET frameworku Socket. Po aktivaci vrstvy se vytvoří asynchronní Accept (přijímací) vlákno, ve kterém po připojení nového klienta proběhne vytvoření dalšího asynchronního vlákna pro další klienty a příjme se uvítací zpráva. Tato zpráva se předá události *OnConnection*, která má jako argumenty objekt třídy, ze které se posílá událost, Socket objekt připojeného klienta a instanci třídy MIMEMEssageParser, které se předá uvítací zpráva k parsování.

Listeners Connection Layer pracuje uplně stejně jako Source Connection Layer, ale s tím rozdílem, že přijímá požadavky od posluchačů a pracuje na portu 3444.

Po připojení klienta a jeho základním zpracování ve spojovací vrstvě se předá handleru serveru, který provádí zbývající připojovací úkony. Uvítací zpráva od klienta musí obsahovat minimálně 4 hlavičky: NGSLogin, NGSPassword, NGSEncoder a NGSBitrate. NGSLogin a NGSPassword slouží pro identifikaci a autorizaci klienta. Na základě těchto přihlašovacích údajů se vyhledává příslušné rádio v databázi (víz. NGSModel) a v případě úspěchu se vrací informace o rádiu. Mezi tyto informace patří jeho jméno, urlkey (identifikační zkratka rádia) a jeho kanály. Urlkey je identifikátor rádio v url adrese (viz. dále), kde stejne jako Login a je jedinečným identifikátorem rádia (buď pro klienta nebo pro posluchače).

NGSEncoder a NGSBitrate jsou identifikátory encodéru, který byl použit pro kompresi dat na klientské straně. Pomocí těchto identifikátoru se v rádiu vytvoří příslušný decodér, který bude příjatá data dekódovat. Obě tyto hlavičky jsou později použité k připojení k rádiu jako posluchač (viz. dále v textu).

Kanály jsou hlavní třídou, která obsluhuje samotné připojené posluchače a posílá jím hotová data. Hlavní třídou pro zpracování kanálů je *RadioChannel*, která dědí ze třídy *BaseSenderBuffer*.

Každý kanál má svou vlastní instanci encodéru, který je mu předán při vytváření. Dále se danému encodéru specifikuje jeho bitrate a počet bloků dat k přednačtení (buffering), než se začnou odesílat posluchačům. Každý kanál si uchovává svou kolekci posluchačů (jejich socketů), kteří jsou na něj připojení.

Při připojení posluchače se prochází kolekce aktivních rádií a jejich kanálů. Pokud se najde shoda, tak se počle odpověď s nastavením daného kanálu. Jako formát odpovědi se používá Shoutcast Metadata protokol a odpověď obsahuje hlavičky ICYResponse, ICYMetaint a ICYBitrate. ICYResponse značí, že se jedná o odpověď, ICYMetaint určuje, v jakém rozsahu budou metadata. V aktuální verzi serveru tato funkce není podporována. ICYBitrate určuje datový tok kanálu. Jelikož je nyní pouze podporován formát MP3, tak není potřeba určovat typ encodéru, ale v případě, že by bylo formátů více, musela by se posílat hlavička Content-Type. Musí se ovšem ověřit, nakolik je tato hlavička podporována u přehrávačů, aby nedocházelo k nekompatibilitě.

K připojení posluchače se musí použít protokol HTTP. Celý formát URL adresy pro připojení je tento (podrobněji v 8.1.4):

```
http://server:3444/radiokey/encoderkey/bitrate
```

- Server značí adresu nebo ip serveru
- Radokey je klíčový identifikátor pro identifikaci rádia (např. myradio)
- Encoderkey určuje formát kanálu, kterém se chceme připojit (např. MP3)
- Bitrate určuje datový tok kanálu (endoréru), ke kterému se chceme připojit. (např. 128)

Takže výsledná URL adresa pro připojení může vypadat např. takto:

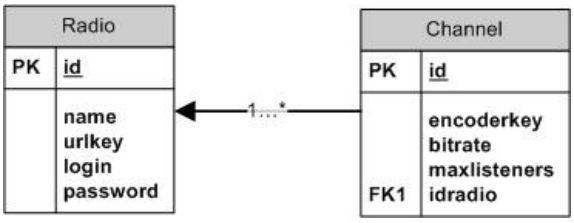
```
http://127.0.0.1:3444/vsb/MP3/128
```

Připojovací vrstva pro posluchače nyní akceptuje MIME uvítací zprávy, které obsahují jako první hlavičku GET nebo OPTIONS ve standardním HTTP formátu
GET /urlkey/encoderkey/bitrate HTTP/1.1. Pokud není nalezena tato hlavička, tak je spojení s posluchačem ukončeno.

Pro ukládání dat využívá server Embeded databázi .NET frameworku SQL Compact 3.5. Práci s databází zajišťuje třída NGSModel. Tato třída obsahuje řadu statických metod pro práci s databází a pro samotné dotazy využívá objektový dotazovací jazyk LINQ. Aby mohl LINQ správně fungovat s SQL Compact databází, musela se nejdříve vygenerovat třída popisující obsah databáze. Jelikož LINQ to SQL podporuje pouze MSSQL databázi, muselo se využít nástroje sqlmetal přímo. Tento nástroj se nachází ve složce s Windows SDK (většinou c:
Program Files
Microsoft SDKs
Windows
v6.0A
bin). Pro samotné vygenerování se použije příkaz:

```
sqlMetal.exe /serialization:Unidirectional N̈GSdb.sdf ̄/code:NGSdb.cs
```

Parametr /serialization nastavuje generování třídy s vlastností serializace tak, že se vytváří pouze jednosměrné spojení. Ve standardním nastavení se vztahy mezi tabulkami generují jako obousměrné a tím se znemožní možnost serializace (viz. RadioHostService). Při jakékoliv změně struktury databáze se musí toto generování provádět znovu. Struktura db lze vidět na obrázku.



Obrázek 14: Struktura databáze

Takto vygenerovaná třída obsahuje databázový kontext, který obsahuje popis tabulek jako tříd a můžeme využít objektových vlastností LINQu.

V průběhu práce a vývoje s takto vygenerovanou třídou jsem narazil na problém, kdy při databazovém dotazu na list rádií jednotlivé objekty neobsahovaly pole kanálů,

i když tyto kanály byly nadefinovány v databázi. Při zkoumání problémů jsem došel až do právě vygenerované třídy, která obsahuje property *public EntitySet¡Channel¿ Channel*, celý její kód vypadá následovně:

```
[Association(Name="Radio_Channel", Storage="_Channel", ThisKey="Id", OtherKey="Idradio",
    DeleteRule="CASCADE")]
[DataMember(Order=6, EmitDefaultValue=false)]
public EntitySet<Channel> Channel
{
   get
   {
        if ((this. serializing
              && (this._Channel.HasLoadedOrAssignedValues == false)))
        {
           return null;
        }
        return this._Channel;
   }
   set
   {
        this._Channel.Assign(value);
   }
}
```

Výpis 8: Kód property EntitySet¡Channel¿ Channel

Při debugování má jsem zjistil, že má property *HasLoadedOrAssignedValues* hodnotu false, i když by měla mít hodnotu true. Různými pokusy, čtením dokumentace a testováním jsem našel celkem jednoduché řešení a funkční řešení pro tento problém. Stačí, přidat malý kousek kódu před podmínku if v get accessoru. Výsledek pak vypadá takto:

```
[Association(Name="Radio_Channel", Storage="_Channel", ThisKey="Id", OtherKey="Idradio",
    DeleteRule="CASCADE")]
[DataMember(Order=6, EmitDefaultValue=false)]
public EntitySet<Channel> Channel
{
   get
   {
        if (this._Channel.IsDeferred)
         this._Channel.Load();

        if ((this. serializing
             && (this._Channel.HasLoadedOrAssignedValues == false)))
        {
           return null;
        }
        return this._Channel;
   }
   set
   {
        this._Channel.Assign(value);
   }
}
```

Výpis 9: Opravený kód property EntitySet¡Channel¿ Channel

Property IsDeferred určuje, jestli je chybí nějaké dotazy ke zpracování. Pokud je hodnota true, tak jesdnoduše na tabulku kanálu zavoláme metodu Load, která nám načte potřebné položky z databáze.

Jednou z vlastností a zároveň problémů LINQu je cachování dotazů. Při vývoji jsem se setkal s problémem, kdy se mi nově přidané kanály pro rádio neukázaly v následném výpisu rádií dokud nebyl server restartován. Problémem bylo právě zmíněné cachování výsledků dotazů a musel jsem hledat řešení. [36]LINQ obsahuje interní metodu ClearCache(), ale k této metodě se nedá hned dostat. Tento problém se dá vyřešit použitím reflexe, kdy si v třídě naší db třídy NGSdb získame metodu ClearCache a následně pomocí invokace ji zavoláme. Celé toto řešení se dá udělat v C# od verze 3 jako extenzní metoda a tím usnadnit následný přístup k této metodě. Za tímto účelem byla vytvořena statická třída NGSModelExtension se statickou metodou ClearCache, která obsahuje výše zmíněný postup.

Protože jsem se snažil víceméně dodržovat MVC model, tak jsem oddělil jednotlivé komponenty od sebe. Modelem je třída NGSModel, která obsluhuje práci s databází. Dále většinu vnitřní logiku obsluhují třídy RadioServer, RadioEntity, RadioChannel a u klientské části RadioClient a SourceRecorder. Pro View byly vytvořené GUI nadstavby, které byly pokud možno co nejvíce oddělené od vnitřní logiky. Proto jsem se rozhodl ke dvoum krokům: k vytvoření konzolové nadstavby serveru, která bude co nejméně náročná na prostředky (aby neubírala výkon na cílovém serveru) a k vytvoření hostovací služby pro komplexní ovládání serveru.

4.4.8 NGSServerGUI

Tato aplikace neprovádí nic jiného, než že spustí WCF službu, která automaticky spustí rádio server. Dále aplikace vypisuje některé informace o událostech na serveru (např. připojení nového klienta a zahájení příjmu dat od něj). Nakonec aplikace obsluhuje události výjimek serveru (NGSException, Exception a HostServiceException), které loguje do souboru. Naneštěstí není tento systém 100% odolný a pokud se vyvolá výjimka na nějakém nepřímo odchytavaném místě, tak aplikace natvrdo havaruje. Tento problém je zapříčiněn kvůli vícevláknovosti a použití více knihoven, kdy se výjimky v nich vyvolané nezachytí příslušnými handlery a zapřičiní pád aplikace. Bohužel ani globální handlery příliš nepomohou a je potřeba vymyslet lepší řešení. Nicméně ty nejdůležitější části jsou víceméně odchytavány a logovány správně.

4.4.9 RadioHostService

Aby bylo možno ovládat a nastavovat server, případně z něj získávat informace, byla vytvořena WCF služba, která je jakýmsi API pro vnější aplikace. Základem služby je rozhraní *ICommandInterface*, které obsahuje všechny nyní používané metody. Aby byla třída funkční jako WPF služba, musí být označena atributem ServiceContract a její metody

jako OperationContract. Poté byla vytvořena třída *CommandInterface*, která implementuje rozhraní ICommandInterface. Největší zajimavostí této třídy jsou události NGSExceptionHandler a ExceptionHandler, které jsou vlastně prodloužením ke stejným událostem, které jsou v třídě RadioServer. Tento způsob byl zvolen právě z důvodu obtížného zachytávání výjimek a jejich předání ke zpracovaní. Takto jsou výjimky v serveru zachyceny a předány do konzolové aplikace, kde jsou zalogovány do souboru.

Celá služba beží na portu 8080 a ke svému nastavení využívá konfigurační soubor. Tento soubor je přítomen u exe aplikace a obsahuje nastavení endpointů, logování chyb a jiné. Služba má vytvořené 2 endpointy:

Prvním endpoitem je tzv. MEX. Tento endpoint slouží k publikování metadat o službě. Tudíž všichni, kdo by jí chtěli využívat, tak si musí stáhnout WSDL informace. Ty obsahují informace o komunikačním rozhraní a mohou být pak použity k vývoji externí aplikace (viz. NGS Server Command Center v kapitole 8.1.3) nebo např. webové stránky publikující aktivní rádia atd.

Druhým endpointem je samotný NGSCI kanál, přes který se už provádí komunikace se serverem. Podporuje např. přídávání a mazání rádií, listování rádií a další.

Samotná komunikace se provádí pomocí HTTP dotazů a zprávy jsou ve formátu SOAP. Díky tomu je služba nezávisla na platformě a může ji využít každý jazyk nebo operační systém, který má implementaci HTTP a SOAP.

5 Budoucí vylepšení

Jako v každém programu, tak v tomto se nabízí celá řada různých vylepšení, která by mohla být pozdější dodělána. Vypustím zde vylepšení typu optimalizace, změna GUI, přidání dalších formátů a protokolů apod. a zaměřím se na ty zajimavější a důležitější.

5.1 Automatický restart

V budoucnu by bylo vhodné vytvořit malou službu, která by hlídala proces rádia. Pokud by náhodou došlo k výpadku ať už z důvody chyby, restartování hostovacího stroje nebo jakýmkoliv jiným způsobem, tak by se provedlo automatické znovuspuštění, aby se minimalizovala doba výpadku. S tímto jde samozřejmě ruku v ruce i obdobná funkce u klienta, který by opakovala pokusy o spojení v případě výpadku.

5.2 Vylepšené logování událostí, výjímek a statistik

Jak bylo zmíněno v textu, tak obě aplikace obsahují základní logování chyb. Bohužel toto logování není úplně 100% a může se stát, že aplikace natvrdo spadne bez zalogování chyby do souboru. Musí se tedy vymyslet lepší systém odchytávání výjimek a tím v případě problémů znát i důvod výjimky.

Další komponentou, která nyní chybí je logování událostí a případných statitik. Log událostí se může využít pro hledání příčin výpadků nebo celkové sledování práce na serveru. Statistiky by se daly využít např. k určování cen reklam apod. V konečném důsledku by jsme získali podrobnější přehled o provozu serveru.

5.3 Advanced Exception Memory Dumper

V případě chyby je důležité vědět příčinu výjimky. K lepší detekci problému se dá využít speciálního Memory Dumperu a provést tzv. otisk paměti, kdy by se celá třída (případně celý proces) včetně hodnot proměnných, stavů apod. uložila do souboru. Poté při zpětné analýze by se daly mnohem lépe najít příčiny chyby a usnadnit tak odstranění.

5.4 Rozhraní pro komunikaci s aplikacema třetích stran

Jedním z problémů, který stále zůstal nevyřešen je příjem metadat od aplikací třetích stran. Pokud např. klient využívá Winamp pro vysílání hudby, tak by se hodilo, kdyby jsme byli schopni nějak získat název skladby, která se aktuálně přehrává. V základu by se dalo využít hookování API funkcí, které otevírají a čtou soubory. Tento systém ovšem nemusí bát přesný, protože spousta aplikací si přednahrává skladby do paměti a nemusíme vždy dostat správný název.

Další možností by bylo vytvořit rozhraní (službu), přes které by se posílaly metadata. Nevýhoda je v tom, že aplikace musí obsahovat plugin, modul nebo jakýkoliv kousek kódu, který by naše rozhraní umělo využít. Pro systém Winamp by se dal naprogramovat vlastní plugin, ale naneštěstí né všechny aplikace mají SDK nebo podporují pluginovatelnost.

5.5 Virtuální zvuková karta

V textu byla zmíněna technologie Virtuálních kabelů. Tato technologie vytvoří virtuální zvukovou kartu, která obsahuje jakoby kabely, přes které můžeme přenášet signál. Celé řešení je sice poměrně silné, ale zároveň i trochu nešikovné, kdy se musí pomoci Audio Repeateru nastavovat různá přesměrování na jiné kabely nebo zařízení, abychom vůbec něco slyšeli.

Vlastní zvuková karta by fungovala na podobném principu. Vytvořil by se ovladač virtuální zvukové karty, který by obsahoval nastavitelné vstupy a výstup. Tyto výstupy by se ovšem přímo přesměrovávaly na fyzické zařízení s tím rozdílem, že by se data na nich kopírovaly na jiné vstupy/výstupy. Tím by jsme dosáhli rozdělení kanálu a mohli bychom např. pomocí Winampu přehrávat a vysílat jednu skladbu a v jiném přehrávači si sami poslouchat jinou, která by nenarušovala vysílání.

5.6 Podpory video vysílání

Implementace celého serveru je velmi univerzální a nyní podporuje pouze zvuk. Není ovšem problém do budoucna přidat podporu pro živé vysílání videa. Díky nynejší implementaci je nutné akorat přidat nové encodérya, hlavičky protokolů a knihovna pro záznam videa např. z webové kamery nebo jiných vstupů a případně kombinovat zvuk i video dohromady.

6 Závěr

Cílem práce bylo vytvořit fungující prototyp internetového rádio serveru, který bude mít základní vlastnosti běžných rádiových serverů. Celý projekt vyžadoval mnoho úsilí a času a byla na něm silně znát doba vývoje, kdy byla aplikace jednou kompletně přepsána a mnohokrát přepisována a upravována. Ze základní verze, která podporovala pouze jedno rádio nakonec vznikl server, který dokáže obsluhovat libovolné množství rádií a jejich kanálů zároveň (v rámci možností fyzického hardwaru). Celý projekt vyžadoval nastudování mnoha různých technologií (jak zvukových, tak síťových), které byly v práci použity včetně těch, které použity nejsou, ale buď jsou plánované nebo již byly zahozeny a nahrazeny vhodnějšími. V průběhu vývoje se vyskytly různé problémy, z nichž většina byla vyřešena, ale některé stále zůstaly nebo nebyly kompletně vyřešeny.

V rámci možností jednoho člověka tedy byla vytvořena aplikace, která dokáže konkurovat i jiným menším aplikacím stejného nebo obdobného typu a osobně jsem s výsledkem spokojen. Navíc je zde mnoho možností na vylepšení a rozšiřování a celý projekt má do budoucna pevné a dobré základy na to, aby se stal silným konkuretnem na poli velkých streamovacích serverů.

Daniel Kapča

7 Reference

[1] Internet Engineering Task Force - http://www.ietf.org

[2] Request for Comments - http://www.ietf.org/rfc.html

[3] GSelector - http://www.gselector.com/

[4] Shoutcast DSP Winamp Plugin - http://www.radioo.cz

[5] Icecast - http://www.icecast.org/

[6] Radioo - http://www.radioo.cz/

[7] Virtual Audio Cable - http://software.muzychenko.net/eng/vac.html

[8] PCM - http://cs.wikipedia.org/wiki/Pulzn%C4%9B_k%C3%B3dov%C3%A1_modulace

[9] LAME - http://lame.sourceforge.net/

[10] GOGO - http://homepage2.nifty.com/kei-i/

[11] Srovnání MP3 převaděčů - http://milaa.tripod.com/mp3Comparison/encoders.html

[12] BASS - http://www.un4seen.com

[13] FMOD - http://www.fmod.org

[14] DirectSound - http://en.wikipedia.org/wiki/DirectSound

[15] DirectX - http://www.microsoft.com/games/en-US/aboutGFW/pages/directx.aspx

[16] XNA - http://www.xna.com/

[17] SlimDX - http://slimdx.org/

[18] VST - http://en.wikipedia.org/wiki/Virtual_Studio_Technology

[19] VST a ASIO SDK - http://www.steinberg.net/en/company/3rd_party_developer.html

[20] ASIO - http://en.wikipedia.org/wiki/Audio_Stream_Input/Output

[21] ASIO4ALL - http://www.asio4all.com/

[22] MIME - http://cs.wikipedia.org/wiki/Multipurpose_Internet_Mail_Extensions

[23] Shoutcast Metadata protocol - http://www.smackfu.com/stuff/programming/shoutcast.html

[24] Shoutcast Metadata protocol 2 - http://forums.radiotoolbox.com/viewtopic.php?t=74

[25] Shoutcast Metadata protocol 3 - http://sphere.sourceforge.net/flik/docs/streaming.html

[26] ID3 Homepage - http://www.id3.org/

[27] ID3 - http://cs.wikipedia.org/wiki/ID3_tag

[28] TagLib - http://developer.novell.com/wiki/index.php/TagLib_Sharp

[29] PLS - http://en.wikipedia.org/wiki/.pls

[30] M3U - http://en.wikipedia.org/wiki/M3U

[31] ASX - http://en.wikipedia.org/wiki/.asx

[32] RTSP - http://en.wikipedia.org/wiki/Real_Time_Streaming_Protocol

[33] MMS - http://en.wikipedia.org/wiki/Microsoft_Media_Server

[34] SDP - http://en.wikipedia.org/wiki/Session_Description_Protocol

[35] RTP - http://en.wikipedia.org/wiki/Real-time_Transport_Protocol

[36] LINQ CACHING - http://blog.robustsoftware.co.uk/2008/11/clearing-cache-of-linq-to-sql.html

8 Přílohy

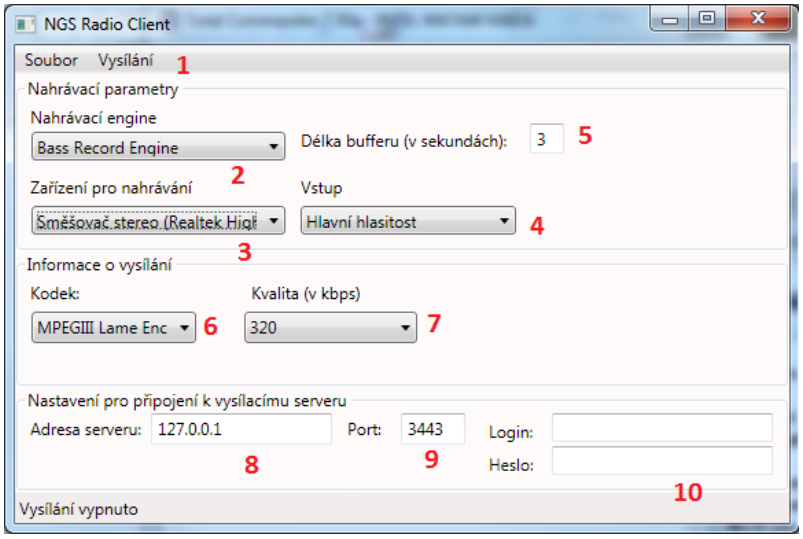
8.1 Příloha A - Uživatelské příručky

Zde se nachází uživatelské příručky všech vytvořených aplikací. Jelikož jsou aplikace postavené na .NET technologiích, tak je potřeba mít nainstalovaný minimálně .NET framework 3.5 SP1. Dále je potřeba mít pro serverovou aplikaci nainstalované běhové knihovny SQL Server COMPACT 3.5 SP1, které nejsou součástí .NET frameworku. Všechny potřebné knihovny a frameworky jsou přítomny na přiloženém CD (viz. kapitola 8.3).

Pro potřeby demonstrace bylo nadefinováno na serveru rádio s těmito údaji (jako ip je zvolen localhost, pokud běží server na jiném počítači, je potřeba změnit adresu nebo ip serveru):

Jméno	VŠB Rádio
Urlkey	vsb
Formát	MP3
Datové toky(kanály)	128
Login	vsb
Heslo	vsb
URL (viz. 8.1.4)	http://127.0.0.1:3444/vsb/MP3/128

8.1.1 Klient



Obrázek 15: Klientská aplikace

Jedná se o klientskou aplikaci, která má na starost nahravání audio dat a jejich následné odeslání na hlavní server. Zde je popis jednotlivých částí aplikace:

1. Menu aplikace.

- Soubor
  - Konec - Ukončí aplikaci
- Vysílání
  - Start - Zahájí nahravání a vysílání rádia
  - Stop - Vypne nahravání a vysílání rádia

2. Nahrávací engine aplikace. Tento engine se použije k záznamu dat. Nyní je pouze přítomen engine postavený na knihovně Bass.
3. Zařízení, které se má použít k záznamu audio dat (např. mikrofón, Stereo Mix - nahravá vše, co jde přes zvukovou kartu).
4. Vstup vybraného zařízení, které se má použít k záznamu audio dat (hlavně využito ve Windows XP).
5. Určuje, kolik sekund se má přednahrát předtím, než se začnou odesílat na server. Slouží jako vyrovnávací paměť. Výchozí hodnout je 3 sekundy. Menší hodnota může způsobit výpadky a přerušení toku dat, naopak velká hodnota způsobí příliš velké zpoždění.
6. Zvukový kodek, který se použije k překodování nahranéných dat.
7. Kvalita překódovaných dat. Větší datový tok znamená větší poslechovou kvalitu, ale větší objem pro přenos dat. Doporučuje se co největší datový tok a tím zajistit nejvyšší kvalitu vysílání.
8. IP adresa serveru, na který se mají nahraná a zpracovaná data odesílat.
9. Port serveru. Standardně 3443.
10. Přihlašovací údaje klientského rádia (Login - uživatelské jméno, Heslo).

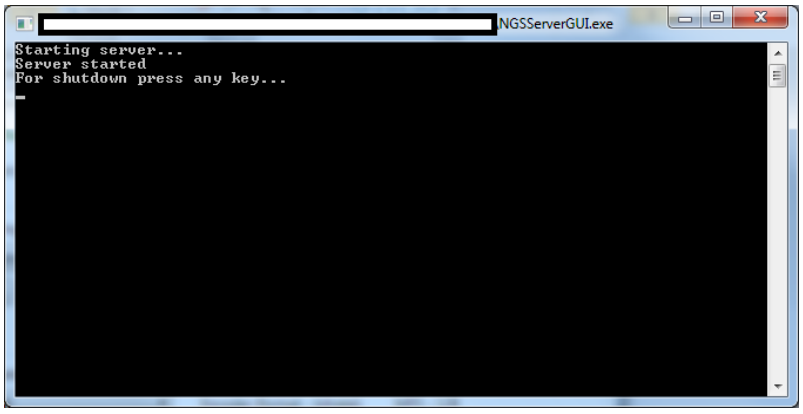
Aby bylo možno nahrávat zvuk, co se momentálně přehrává na počítači, je potřeba mít vybrané jako nahrávací zařízení **Stereo Mix** nebo **What you Hear**. Pokud toto zařízení není přítomné, tak je možné, že je pouze vypnuté a v ovladacích panelech je možné jej zapnout.

Aby bylo zajištěna co nejvyšší vysílací kvalita, je dobré vysílat v co nejvyšší kvalitě, protože server provádí automatický převod na nižší datové toky (podle nastavení kanálů). Pokud bychom vysílali v nižší kvalitě, ale na serveru měli kanály, které by převádely data na vyšší datový tok (než v jakém vysíláme), tak by se kvalita signálu nezlepšila a navíc by se dramaticky zvýšil datový objem potřebný pro přenos dat k posluchačům.

Z důvodu zabezpečení byla zavedena nutnost přihlašování. K přihlášení stačí momentálně pouze přihlašovací jméno a heslo, pomocí kterého se identifikuje správné rádio a vytvoří se vysílací sloty. Přihlašovací údaje se neukládají a musí být vždy zadány znovu.

Pro správný chod je vyžadováno povolení zápisu do Temp složky uživatelského účtu!!!

8.1.2 Server



Obrázek 16: Serverová aplikace

Jedná se o hlavní serverovou aplikaci, která zajišťuje příjem audio signálu od zdrojových klientů (rádio stanic), jejich překodování na nadefinované formáty a datové toky a následnou distribuci mezi připojené posluchače. Celé rádio dokáže zpracovat neomezené množství rádií a jejich kanálů (popis v kapitole 8.1.3) a jediným omezením je fyzický hardware.

Celý server je jedna konzolová aplikace, která spustí WCF službu na portu 8080 a vytvoří 2 tzv. Endpointy (přípojné body). První Endpoint slouží k nabídce metadat WCF služby (MEX) a druhý k samotnému poskytnutí ovladacího rozhraní (NGSCI).

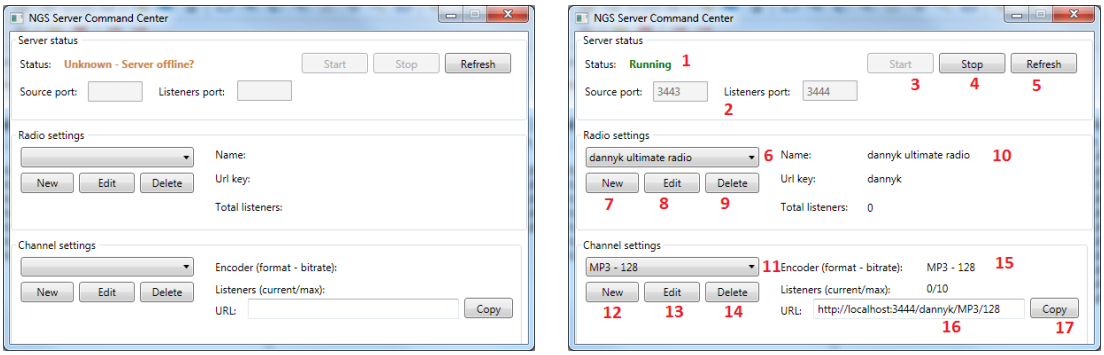
Aby server byl co nejméně náročný na prostředky, tak mu nebylo tvořeno žádné významnější grafické GUI. Stávající konzolové okno pouze informuje o spuštění WCF služby a o zahájení a ukončení příjmu dat jednotlivých klientských rádio stanic.

Veškerá nastavení WCF služby se nachází v souboru *NGSServerGUI.config* a při změně je potřeba server restartovat. Nyní nejdůležitější nastavení:

httpGetEnabled	Povolení publikování metadat skrze http protokol
includeExceptionDetailInFaults	Povolení odeslání rozšířené informace o chybě klientské aplikaci
baseAddress	Náslouchací adresa služby

Pro správný chod je vyžadováno povolení zápisu do Temp složky uživatelského účtu!!!

8.1.3 NGS Server Command Center



(a) Ovládací panel serveru - nepřipojen (b) Ovládací panel serveru

Obrázek 17: Ovládací panel serveru

Jelikož samotný server z důvodu úspory systémových prostředků neobsahuje uživatelské rozhraní a tím prakticky žádnou přímou možnost konfigurace rádií, kanálů a vlastností, tak bylo potřeba vytvořit uživatelskou nádstavbu využívající vytvořené WCF služby. Nyní si popíšeme hlavní části aplikace:

1. Stav serveru.
2. Nastavení portu, na kterých se má serveru spustit.
3. Spustí rádio server.
4. Vypne rádio server.
5. Obnoví status rádia a seznamy rádií a kanálů.
6. Seznam nadefinovaných rádií.
7. Dialog pro nadefinování nového rádia.
8. Dialog pro editaci již nadefinovaného rádia.
9. Smaže definici rádia.
10. Základní informace o rádiu.
11. Nadefinované kanály rádia.
12. Dialog pro definici nového kanálu rádia.
13. Dialog pro editaci již nadefinovaného rádia.
14. Smaže definici kanálu.

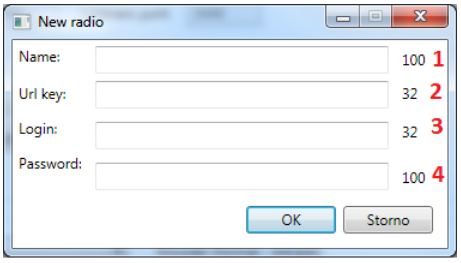
15. Základní informace o kanálu.
16. URL adresa kanálu rádia.
17. Okopíruje URL adresu do schránky.

Stav serveru může nábývat hodnot Running, Stopped a Unknown. Unknown stav znamená, že se aplikace nemohla připojit k serveru a pravděpodobně to znamená, že je server vypnutý. Obnovování se neprovádí automaticky.

Server podporuje nastavení portů, na kterých bude naslouchat. Source port je určen klienstkým rádio stanicím a Listeners port posluchačům. Pokud byl změněný jeden z portů, je možné opět nastavit port výchozí nastavením výchozího portu (source = 3443, listeners = 3444) nebo zadaním portu 0 (nula).

Důležitým tlačítkem je Refresh, který obnoví status rádia a seznamy rádií a jejich kanálů. Rádia i kanály ukazují základní informace o nastavení rádií a jejich kanálů. U kanálů je možné se podívat na výslednou URL adresu pro připojení posluchačů a lze ji přímo pomocí tlačítka Copy okopírovat do schránky.

Stejně jako server, tak i tato aplikace má konfigurační soubor *NGS Server Command Center.config*, který obsahuje nastavení klinta pro připojení k WCF službě serveru. Nejdůležitějším atributem je zde **address** u endpoint tagu, který určuje adresu serveru, na který se služba připojí, a kde očekává WCF službu serveru.
Nyní si popíšeme dialogy pro definování rádií a kanálů:

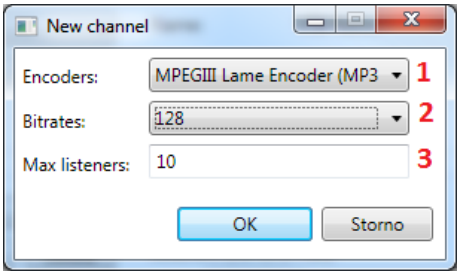


Obrázek 18: Dialog nového rádia

1. Jméno rádia.
2. URL identifikátor, kterým se rozliší požadované rádio v url adrese.
3. Uživatelské jméno rádia
4. Heslo rádia

Čísla vedle zadávacích polí určují počet zbývajících znaků.
Url identifikátor a Login musí být jedinečné.

URL identifikátor je jedna z nejdůležitějších částí. Slouží jako identifikační část v url při připojení posluchače a identifikuje požadované rádio (viz. část 8.1.4).



Obrázek 19: Dialog nového kanálu

1. Formát kanálu, do kterého se překódují data od klienta.
2. Datový tok encodéru.
3. Maximální počet posluchačů.

Položka Encoder a Bitrate slouží stejně jako Urlkey k identifikaci, ale tentokrát kanálu pomocí url. (viz. část 8.1.4).

8.1.4 Formát URL

Již několikrát byl v textu zmíněn formát URL. V této části je přesně popsáno, co je tím myšleno.

Server podporuje pro připojení posluchače protokol HTTP, který automaticky posílá GET požadavek obsahující požadovanou URL (soubor). Ukázku takové GET zprávy lze vidět na výpise 3. Tato url je pak použita k identifikaci požadovaného rádia a kanálu. Celá URL může ve výsledku vypadat takto:

http://server:port/radiokey/encoderkey/bitrate

server	IP nebo adresa serveru
port	Port, na kterém naslouchá server posluchačům
radiokey	Identifikátor rádia
encoderkey	Identifikátor encodéru (kanálu)
bitrate	Identifikátor datového toku encodéru (kanálu)

Serverový identifikátor a port není potřeba dále rozebírat, jelikož na nich není nic neobyčejného. Radiokey je identifikátor rádia. Každé rádio má totiž svůj název, který často bývá dlouhý a obsahuje mezery nebo diakritiku, proto bylo nutné zavést nejaký jedinečný identifikátor, podle kterého by se dalo rádio nalézt právě pomocí URL. Obdobné je to u Encoderkey, kdy opět jakýkoliv podporovaný převaděč má svůj název a tzv. ShortName, který určuje nějčastěji zkratku jeho formátu (jako např. MP3). Jako poslední identifikátor slouží datový tok, který určuje kvalitu kanálu.

Ve výsledku probíhá celý proces identifikace tak, že se nejdříve pomocí Radiokey dohledá rádio. V případě úspěchu se hledá kanál, který odpovídá požadovanému formátu a kvalitě (datovému toku = bitrate). Pokud se nalezne takový kanál, tak se zahájí odesílání dat posluchači v požadovaném formátu a kvalitě u požadovaného rádia.

8.2 Příloha B - Struktura zdrojových kódů

Zdrojové kódy jsou rozdělené na 2 složky (hlavní projekty): NGS Radio Server a NGS Server Command Center. Projekt se serverem obsahuje řadu podprojektů, kdy každý z nich po úspěšné kompilaci kopíruje svoje zkompilované soubory do složky Output/{Client/Server}/Debug (Release verze nemá nastaveny Post-build příkazy).

Většina podprojektů využívá různé vedlejší knihovny, které byly umístěny buď do složky *libs* nebo do výstupní Build složky.

8.3 Příloha C - Obsah CD

- Bin - Spustitelné soubory serveru, klienta a command centera
- Ostatní - Různé soubory (manuál k lame, zdrjové kódy jiných encodérů)
- Programátorská dokumentace - Dokumentace ke zdrojovým kódům
- Redist - Runtime knihovny potřebné pro spuštění spustitelných souborů
- Text - Text diplomové práce v pdf
- Videa - Tutoriálové videa pro základní práci s aplikacema
- Zdrojové kódy - Zdrojové kódy diplomové práce